НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

До захисту допущено:
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
(підпис) (вл.ім'я, прізвище)

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: « Система підтримки навчання студентів – учасників європейських програм мобільності »

Виконав : студент *IV* курсу, групи ІС-63

Фам Суан Хоанг _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доц., к.т.н., доц. Тєлишева Тамара Олексіївна _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації доц., к.т.н., доц. Тєлишева Тамара Олексіївна _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент доц., к.т.н., доц. Тєлишева Тамара Олексіївна _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматизованих систем обробки інформації та управління
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

(підпис) (вл.ім’я, прізвище)

ЗАВДАННЯ
на дипломний проєкт студенту

Фам Суан Хоанг
(прізвище, ім’я, по батькові)

1. Тема проєкту «Система підтримки навчання студентів – учасників європейських програм мобільності »

керівник проєкту доц., к.т.н., доц. Тєлишева Тамара Олексіївна
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “7”травня 2020 р. №1081-с

2. Термін подання студентом проєкту “01”червня 2020 року

3. Вихідні дані до проєкту

Технічне завдання, програмне забезпечення, графічний документ, пояснювальна записка

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі
2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна варіантів використань

2. Схема бази даних

3. Схема структурна класів програмного забезпечення

4. Схема діаграма станів

5. Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*15.04.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*25.04.2020*	
3.	*Постановка та формалізація задачі*	*26.04.2020*	
4.	*Розробка інформаційного забезпечення*	*28.04.2020*	
5.	*Алгоритмізація задачі*	*01.05.2020*	
6.	*Обґрунтування використовуваних технічних засобів*	*01.05.2020*	
7.	*Розробка програмного забезпечення*	*02.05.2020*	
8.	*Налагодження програми*	*09.05.2020*	
9.	*Виконання графічних документів*	*09.05.2020*	
10.	*Оформлення пояснювальної записки*	*10.05.2020*	
11.	*Подання ДП на попередній захист*	*15.05.2020*	
12.	*Подання ДП на основний захист*	*01.06.2020*	
13.	*Подання ДП рецензенту*	*02.06.2020*	

Студент Фам Суан Хоанг

Керівник Тамара Тєлишева

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6324.00.000 ПЗ	Пояснювальна записка	99	
3	А4	ДП 6324.01.000 ТЗ	Технічне завдання	8	
4	А3	ДП 6324.02.000 ССВ	Схема структурна варіантів використань	1	
5	А3	ДП 6324.03.000 ССС	Схема структурна станів системи	1	
6	А3	ДП 6324.04.000 СБД	Схема бази даних	1	
7	А3	ДП 6324.05.000 ССК	Схема структурна класів програмного	1	
			забезпечення		
9	А3		Рішення з математичного забезпечення	1	
10	А3	ДП 6324.06.000 КЕ	Креслення вигляду екранних форм	3	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6324 00.000.00		
Розробн.		Фам С. Х.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Тєлишева Т.О.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-63	
Н/контр.		Проскура С.Л.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: Система підтримки навчання студентів – учасників європейських програм мобільності.

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з шести розділів, містить 30 рисунків, 10 таблиць, 1 додатків, 9джерел.

Дипломний проект присвячений розробці процесу подання документів для студентів – учасників європейських програм мобільності.

Основна мета проекту- зменшити втрати часу в процесі прийняття участі в програмах академічної мобільності за рахунок використання он-лайн технології оформлення документів.

У розділі інформаційного забезпечення показує вхідні, вихідні дані та структура бази даних проекту.

Розділ математичного забезпечення присвячений методувизначенняфактора, який має найбільший вплив на загальному рейтингу конкурсу.

Програмне забезпечення показує засоби розробки та основні вимоги до технічного забезпечення та побудував діаграмкласів, послідовність та компонентів, показувати специфікації функцій.

У технологічному розділі показує керівництво коритувача, було детально описано як працювати з формою та було проведено випробування основних функціональностей.

ПРОГРАМ МОБІЛЬНОСТІ, ДОКУМЕНТИ, ВІДДУЛ МОБІЛЬНОСТІ, КООРДИНАТОР, ІНДИВІДУАЛЬНИЙ НАВЧАЛЬНИЙ ПЛАН, ЄКТС, РЕЙТИНГ КОНКУРСУ

					ДП 6324.00.000 ПЗ		
Зм.	Арк.	Прізвище	Підпис	Дат			
Розроб.		Фам Суан Хоанг.			Система підтримки навчання студентів – учасників європейських програм мобільності.	Літ.	Лист 99 / Листів
Перевірив.		Тєлишева Т. О.					
Н. кон.		Проскура С. Л.				КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-63	
Затв.		Павлов О. А.					

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of six sections, consists 29 images, 10 tables, 1 applications, 9 sources.

The diploma project is devoted to the development of the process of submitting documents for students - participants of European mobility programs.

The main goal of the project is to reduce the loss of time in the process of participating in academic mobility programs by using online technology for paperworking.

The information support section shows the input, output and structure of the project database.

The section of mathematics is devoted to method of determining the factor that has the greatest influence on the overall ranking of the competition

Software sectionshowsthe development tools and basic hardware requirements and has built class diagrams, sequences and components and show the specifications of functions.

In the technological section shows the user's guide, described in detail how to work with the form and tested the basic functionalities.

MOBILITY PROGRAMS, DOCUMENT, MOBILITY DEPARTMENT, COORDINATOR, INDIVIDUAL CURRICULUM, ECTS, RANKING OF COMPETITION

ЗМІСТ

ВСТУП

ЕРАЗМУС+ – це програма Європейського Союзу на період 2014-2020 рр., що підтримує проекти співпраці, партнерства, заходи і мобільність у сфері освіти, професійної підготовки, молоді та спорту. Разом з понад 150 країнами світу, Україна є однією з країн-партнерів (Partner Countries) Програми Еразмус+.[1]

Актуальність теми. Але процес пошукууніверситету-партнера, який відповідає категорії студента і процес подачі документів витрачають багато часу. Тому тема дипломного проекту, в якому розглядаються питання підтримки студента - учасника програми академічної мобільності, є актуальною.

Практичне значення одержаних результатів очевидно, тому що процеси подання і оформлення документів мають велике значення для організації академічної мобільності.

Програми обмінів передбачають подачу пакета документів, таких як заяви, договори, довідки, індивідуальний навчальний план, сертифікат про рівень іноземної мови та ін. Кредитна мобільність передбачає розуміння що є системою кредитів (ЄКТС) і як здійснюється накопичення та трансфер кредитів, що орієнтований на особу, яка навчається, і заснована на принципах прозорості процесів навчання, викладання та оцінювання.

Оскільки програма обміну проводиться в навчальному році, це означає що учасник програми фактично пропускає всі лекційні, практичні заняття, контрольні роботи. Для того, щоб учасники програми не були відраховані з навчального закладу, вони повинні заповнити і здати два важливі документи: це план навчання Learning Agreement (LA) та індивідуальний навчальний план (ІНП).

План навчання LA - це список предметів, яким хоче навчитися учасник в університеті – партнері. Так як програми навчання в університеті – партнері і нашому університеті є різними, то учасник повинен вибирати частину таких курси, за якими можуть зарахувати кредити в своєму університеті після закінчення програми обміну, а друга частина кредитів – курси, які він вибирає за своїм бажанням. Кількість кредитів і зміст курсів, вибраних для зарахування повинні відповідати вимогам програми своєго університету. Через те, що важко з`орієнтуватися в відповідності змістів курсів університетів – партнерів і свого університету без детальної інформації на англійскій мові про навчальні програми курсів , студент змушений здавати повну сесію, як в вузі – партнері, так і в своему вузі після закінчення програми обміну. В індивідуальному навчальному плані він має вказатикурси за семестром і терміни здачі всіх заліків і іспитів.Це важке навантаження,які не всі успішно долають.

Дипломний проект присвячений розробці системи, яка б підтримувала учасника конкурса на мобільність в процесі подання документів для програми обміну і сприяла правильгому вибору кредитів.

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Опис предметного середовища

Зазвичай, у кожному році два рази відбувається конкурс студентів для програми обміну , а саме в весняному і в літньому семестрах. Короткострокові програми обміну: кредитна мобільність на навчання (3-12 міс.) та на практику (2-12 міс.) для українських студентів бакалаврів, магістрів, аспірантів, докторантів [5].

Для того, щоб брати участь в програмі, кандидат повинен бути громадянином України , володіти англійською мовою на рівні B1, крім цього, участь можуть брати студенти від другого року навчання і студент може взяти участь в програмі тільки 1 раз. За умовами Еразмус+ КА1 студент може отримувати стипендію або вчитися без стипендії, але за навчання в іноземному ВНЗ студент не платить. Сума стипендії залежить від ліміту, який визначається кожною країною[1].

Студент має подати заяву для участі в конкурсі та підготувати необхідні документи англійською мовою.Документи подаються в ручному режимі,що забирає багато часу, іноді потрібно щось переробляти і знов оформляти документи.

1.1.1. Опис процесу діяльності

На даномуетапіпроцесс подачі документів можна описати діаграмами IDEFO на рисунках 1.1, 1.2

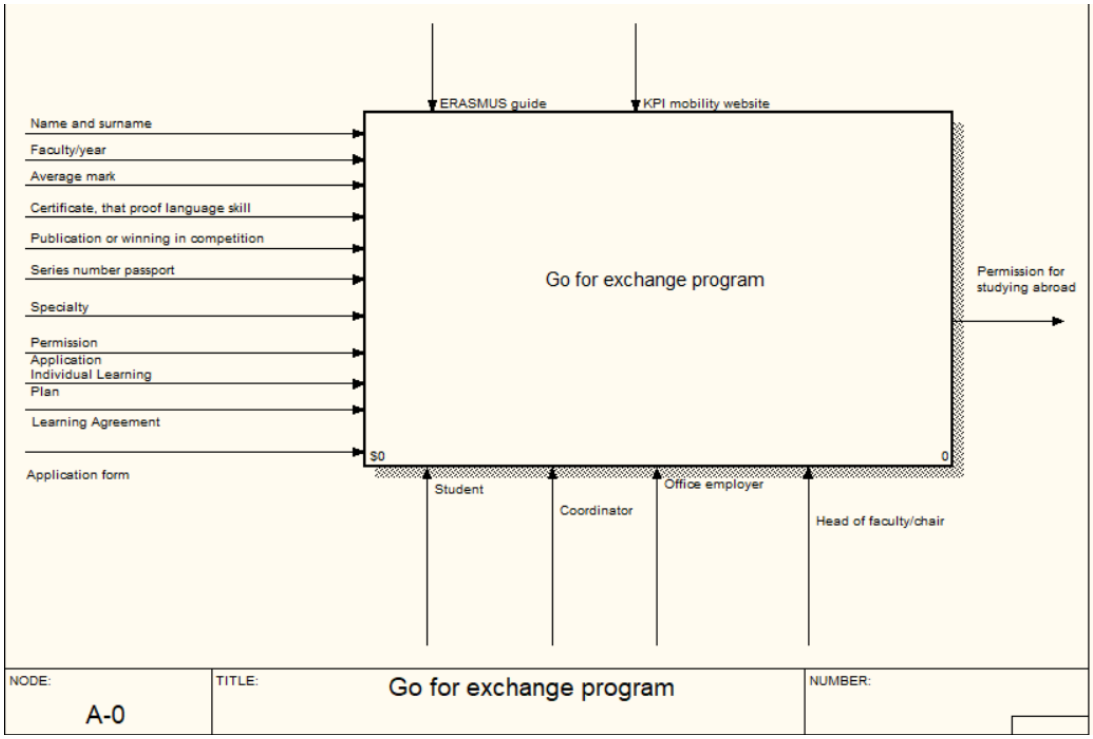


Рисунок1.1 — Модель бізнес-процесу участі в конкурсі на мобільність

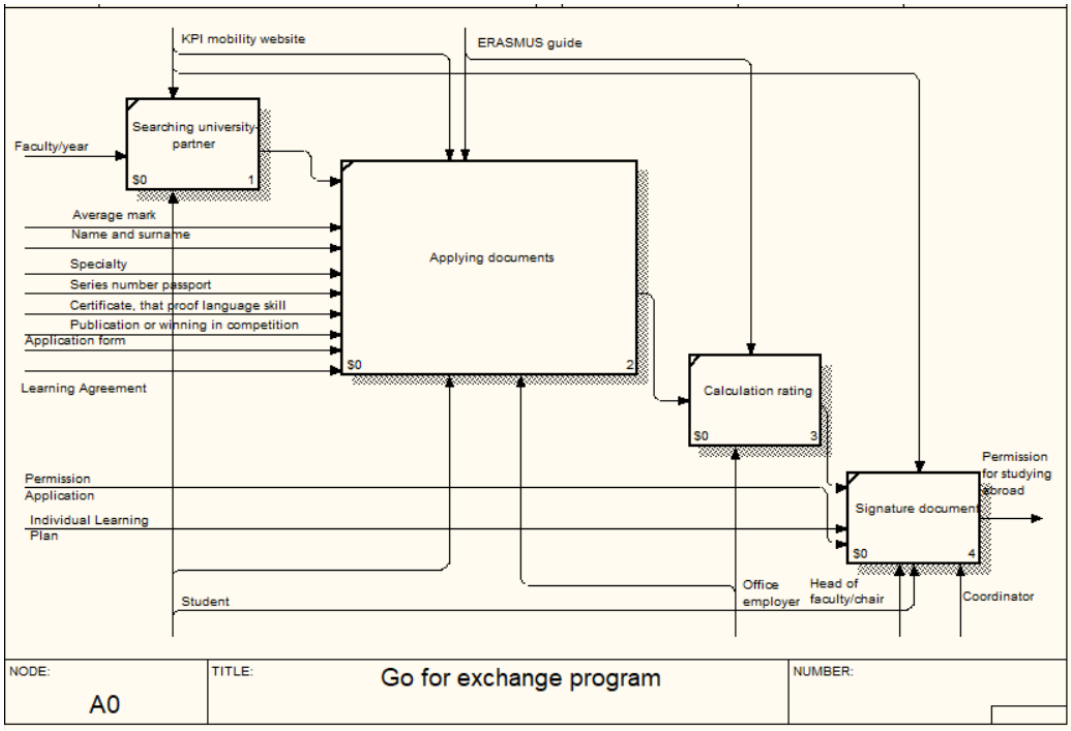


Рисунок 2.2 — Декомпозиція бізнес-процесу участі в конкурсі на мобільність

Процес пошуку виш-партнера, який відповідає вимогам і умовам учасників відбувається досить складно Учасник повинен переглянути всіх виш-партнерів, які є у списку на сайті відділу академічної мобільностіі перевірити чи підходять вони для його спеціальності та за періодом навчання, оскільки для кожного виш-партнера існують певні вимоги і програми.

Процес подачі документів відбувається в ручному вигляді, це означає, що студент заповнює заяву-анкету, LearningAgreement вручну і приносить всі ці документи до відділу мобільності. Всі подані документи, важливі критерії для обчислення рейтингу є в паперовому вигляді. Працівники відділу мобільності втрачають багато часу для обчислення рейтингу окремого учасника. Звичайно, цей процес триває від 3 - 4 тижнів.

Після конкурсного відбору переможець конкурсу переходить на наступний етап. Учасник повинен подати декану заяву на індивідуальне навчання і індивідуальний навчальний план (ІНП) після візи від координатора академічної мобільності факультету і завідувача кафедри. Удеяких ситуаціях вони є відсутніми в університеті, тому учасник повинен чекати день або тиждень, щоб отримати підпис.

В системі пропонується автоматизація пошуку виш-партнера, який відповідає категорії студентів і пропонує вибір,що скорочує втрати часу.

Система надає учаснику форму для заповнення анкети і зберігає ці дані в базі даних. Учаснику вже не треба приносити паперові документи до відділу мобільності.

Система автоматично підраховує рейтинг конкурсанта, враховуючи всі дані комплексно, таким чином скорочується час на визначення кандидата.

Після наказа на мобільність система надає форми заяви і договора та індивідуального плану в електронному вигляді на візування та затвердження декану, що скорочує час оформлення кінцевого етапу.

1.1.2. Опис функціональної моделі

В процесі участі в конкурсі є 4 типи акторів які будуть взаємодіяти з системою:

- учасники конкурсу - студенти, викладачі;
- працівники відділу мобільності - вони отримують документи від учасників і виконують операцію обчислення рейтингу, дають відповідь на всі питання, які пов'язані з програмою обміну;
- координатор - відповідає за програми обмінів в кожному факультеті;
- завідувач кафедри /декан факультету - затверджує договір з учасником мобільності про індивідуальний план навчання .

На рисунку 1.3 показана діаграма варіантів використання системи.

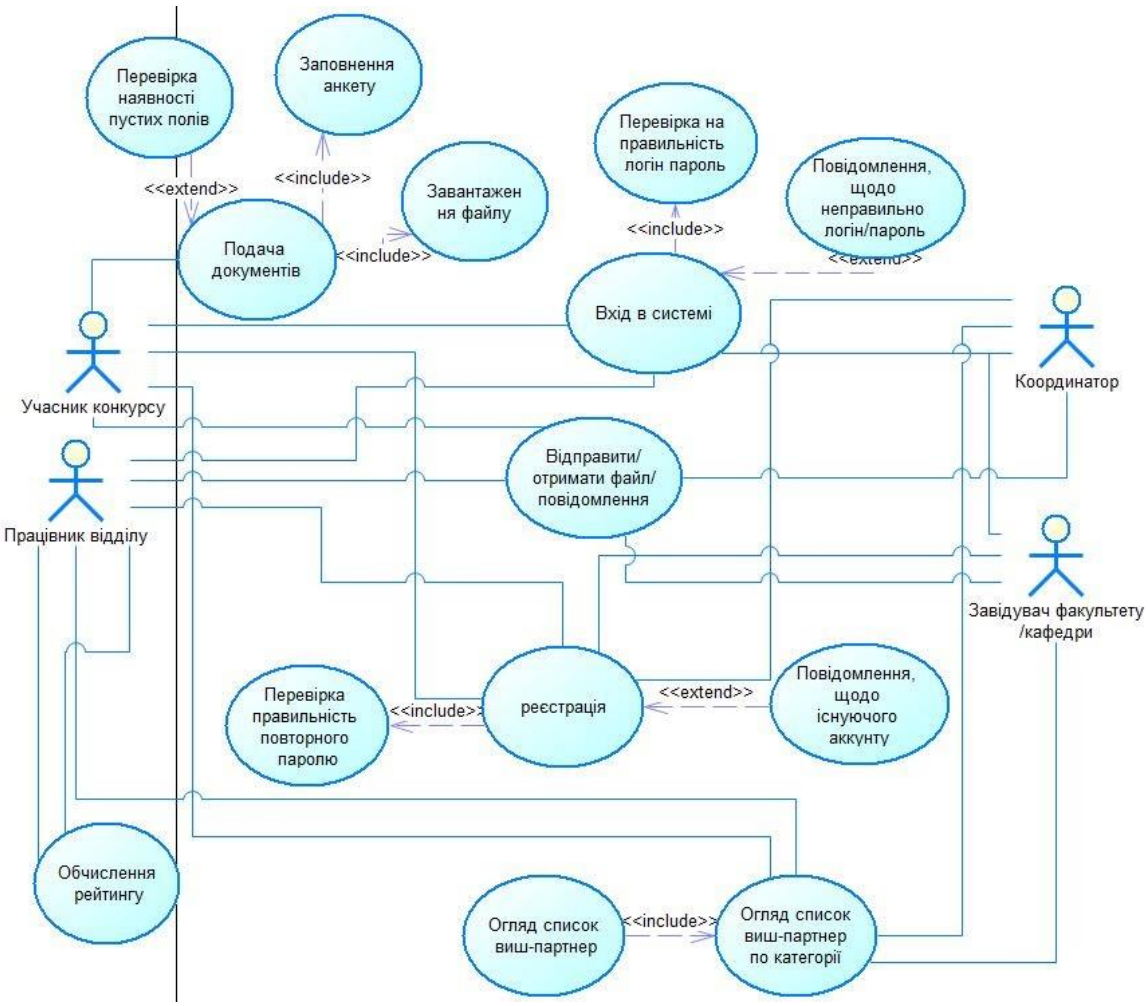


Рисунок3.3 — Діаграма варіантів використання системи

1.2. Огляд наявних аналогів

Портал для іноземних студентів університету Ройтлінгена- https://hsreutl.moveon4.de/locallogin/5538c076140ba02636000000/eng#.

Для входу до системи, ви отримаєте посилання для реєстрації від координатора з університета-партнера. Всі процеси подачі документів відбуваються в он - лайн режимі. Система оформляє бланк із заповненою інформацією, після перевірки бланк буде відправлений до координатора і ви отримаєте копію. На даному кроці ви вже завершили 80% процесу подачі документів і вже можете перейти до наступного кроку – подачі документів на поселення. В порівнянні з цим порталом, майже всі функціонали подібні до запропонованої системи.

Єдина різниця – запропонована в проекті система дає можливість прорахувати рейтинг учасників у конкурсі. Вхідна сторінка цього порталу показана на рисунку 1.4.

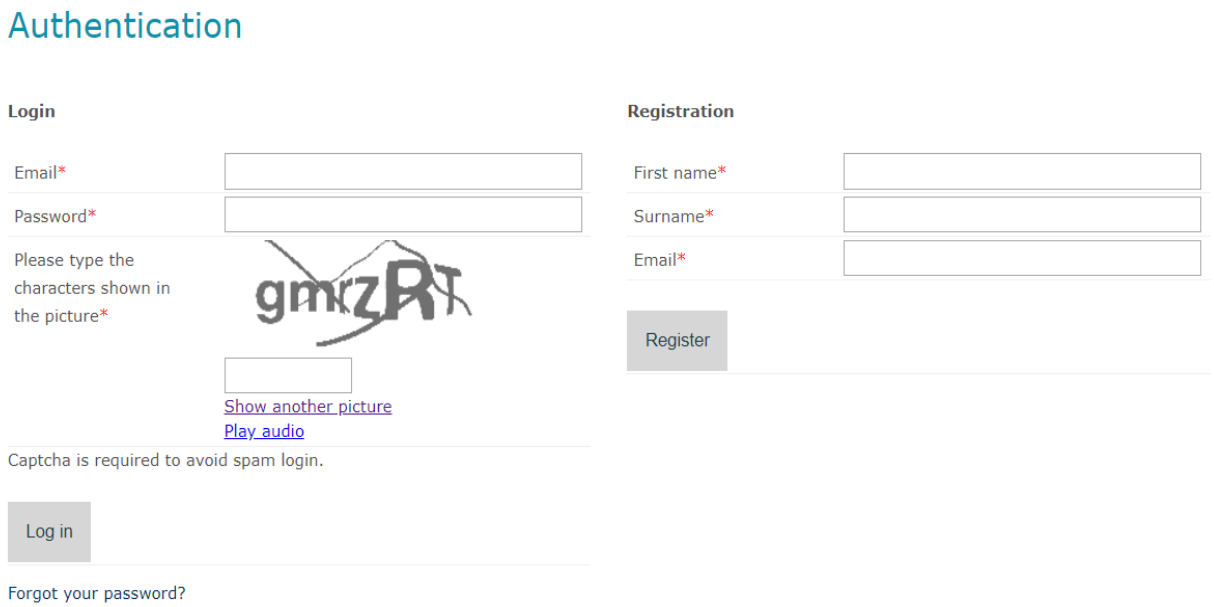


Рисунок3.4 — Сторінка авторизації сайту університету Ройтлінгена

1.3. Постановка задачі

1.3.1. Призначення розробки

Розробити систему, яка дає студентам можливість шукати університет-партнера відповідно до категорії студента і дає можливість надіслати заяви та ІНП до координатора, відділу мобільності, завідуючого кафедри/ деканату на затвердження.

1.3.2. Цілі та задачі розробки

Основна мета проекту- зменшити кількість часу в процесі прийняття участі в програмі академічної мобільності за рахунок використання он-лайн технології оформлення документів .

Для досягнення цілі необхідно вирішити такі задачі:

- провести аналіз процесів оформлення документів;
- розробити структуру та функціональну модель он-лайн технології оформлення документів;

- визначити інформаційне та розробити програмне забезпечення для технології;
- провести тестування технології та експериментальне впровадження.

Висновок до розділу

В цьому розділі було обґрунтовано доцільність розробки системи підтримки навчання студентів - учасників програм європейських обмінів. Було сформульовано ціль та задачі дипломного проекту, наведеніі існуючі аналоги системи.

Також описані існуючі бізнес-процеси проведення документів на конкурс та оформлення ІНП в діаграмах IDEFO та розроблено функціональну модель системи підтримки учасників конкурса, описані дії акторів.

2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Вхідні дані

В вхідних даних для роботи системи студент повинен вказати наступні параметри:

- ПІБ;
- назва факультету;
- назва кафедри;
- курс.

Для участі в конкурсі учасник повинен вказати наступні параметри:

- серію та номер закордонного паспорту, аддресу електронної пошти;
- напрям підготовки;
- рівень вищої освіти;
- назву виш-партнера;
- середній бал;
- середній бал по спеціальності;
- інформацію про сертифікат, який засвідчує рівень знання іноземної мови;
- документи, які засвідчують наявність публікацій, наукові-дослідницьку роботу, участь в олімпіадах;

2.2. Вихідні дані

Вихідними даними роботи системи є рейтинг учасника конкурса.

Опис структури бази даних

Для розробки бази даних використаний SQL Server. База даних містить 6 таблиць, які наведені в таблицях 2.1, 2.2.

Таблиця 2.1 – перелік сутностей в бази даних

№	Назва таблиці	Сутність
1	Exchange	Список програм обмінів
2	Users	Таблиця користувачів системи
3	Faculty	Факультет
4	Chair	Кафедра
5	App_Docs	Заяви на конкурсу
6	Doument	Список поданих документів

В таблиці 2.2 було наведено детальний опис сутностей в базі даних

Таблиця 2.2 – Опис сутностей в базі даних

Назва таблиці	Назва стовпця	Тип даних	Опис поля
Exchange	id	int	Первинний ключ
	program	varchar(100)	Назва виш-партнер
	country	varchar(100)	Країна виш-партнера
	faculty_id	int	Посилання до таблиці Факультету
	program_year	int	Вимог курс для учасників
Faculty	id	int	Первинний ключ
	faculty_name	varchar(100)	Назва факультету
	coordinator	varchar(100)	Ім’я координатора
	head_faculty	varchar(100)	Ім’я завідувач факультету
Chair	id	int	Первинний ключ
	chair_name	varchar(100)	Назва кафедри
	head_chair	varchar(100)	Ім’я завідувач кафедри
	faculty_id	int	Посилання до таблиці Факультет

Продовження таблиці 2.2

Назва таблиці	Назва стовпця	Тип даних	Опис поля
Users	id	int	Первинний ключ
	username	varchar(100)	ПІБ користувача
	userlogin	varchar(100)	Логін користувача
	faculty_id	int	Посилання до таблиці Факультету
	chair_id	int	Посилання до таблиці Кафедри
	user_year	int	Курс
	user_role	varchar(100)	Позиція користувач
	user_password	varchar(100)	Пароль користувач
Doument	id	int	Первинний ключ
	id_student	int	Посилання до таблиці Users
	application	varchar(100)	Назва файл заяви
	invitation	varchar(100)	Назва файлу запрошення
	invitation_trans	varchar(100)	Назва файлуперекладу запрошення
	inp	varchar(100)	Назва файлу ІНП
	faculty_id	int	Посилання до таблиці Faculty
	nakaz	varchar(100)	Назва файлу наказу
	contract	varchar(100)	Назва файл договору

Продовження таблиці 2.2

Назва таблиці	Назва стовпця	Тип даних	Опис поля
App_Docs	id	int	Первинний ключ
	id_program	int	Посилання до таблиці Exchange
	id_student	int	Посилання до таблиці Users
	passport_number	varchar(100)	Серія і номер паспорту
	passport_scan	varchar(100)	Посилання до скану паспорту
	english_skill	varchar(100)	Рівень володіння мови
	type_certificate	varchar(100)	Тип сертифікату
	certificate_scan	varchar(100)	Посилання до скану сертифікату
	publication_avail	int	наявність публікації, статті
	publication_file	varchar(100)	Назва файлу публікації
	competition_avail	int	наявність перемоги в конкурсу
	competition_file	varchar(100)	Назва файлу сертифікатів конкурсу
	learning_agreement_file	varchar(100)	Посилання до файлу план навчання
	email	varchar(100)	Адрес електронної пошти
	avatar	varchar(100)	Посилання до фото учасника
	publication_score	int	Бал за публікації
	competition_score	int	Бал за конкурсу
	rating	float	Загалний рейтинг

Виходячи з таблиць 2.2, розроблено діаграма бази даних, яка показано на рисунку 2.1

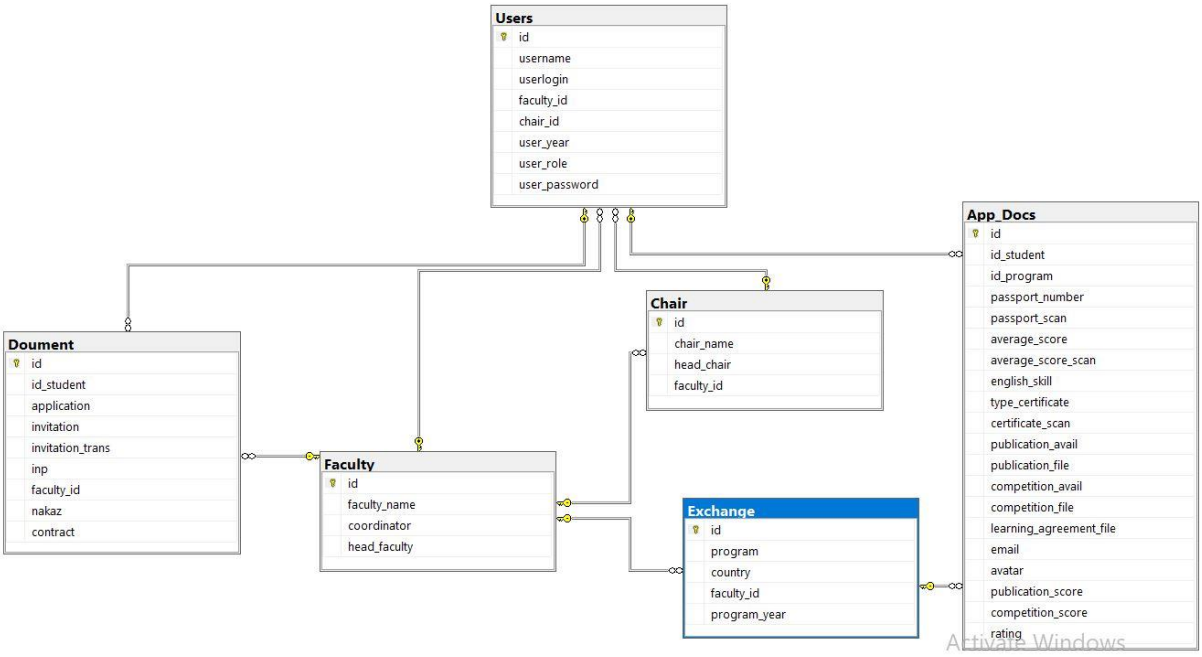


Рисунок 2.1 — Діаграма бази даних системи

Для участі в конкурсі в університеті необхідно надати документи, які зазначені на рисунки 2.2

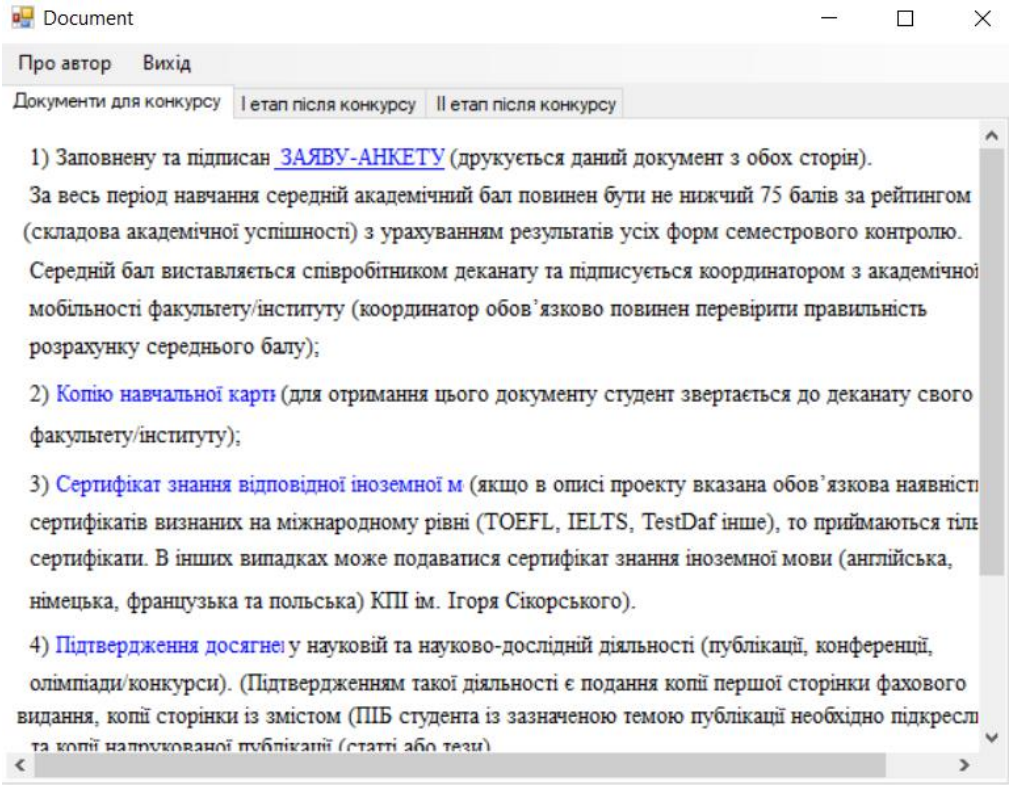


Рисунок2.2 — Форми основних документів для участі в конкурсі [2]

Основні документи,які необхідно подати після конкурсу переможцями наведено на рисунку 2.3

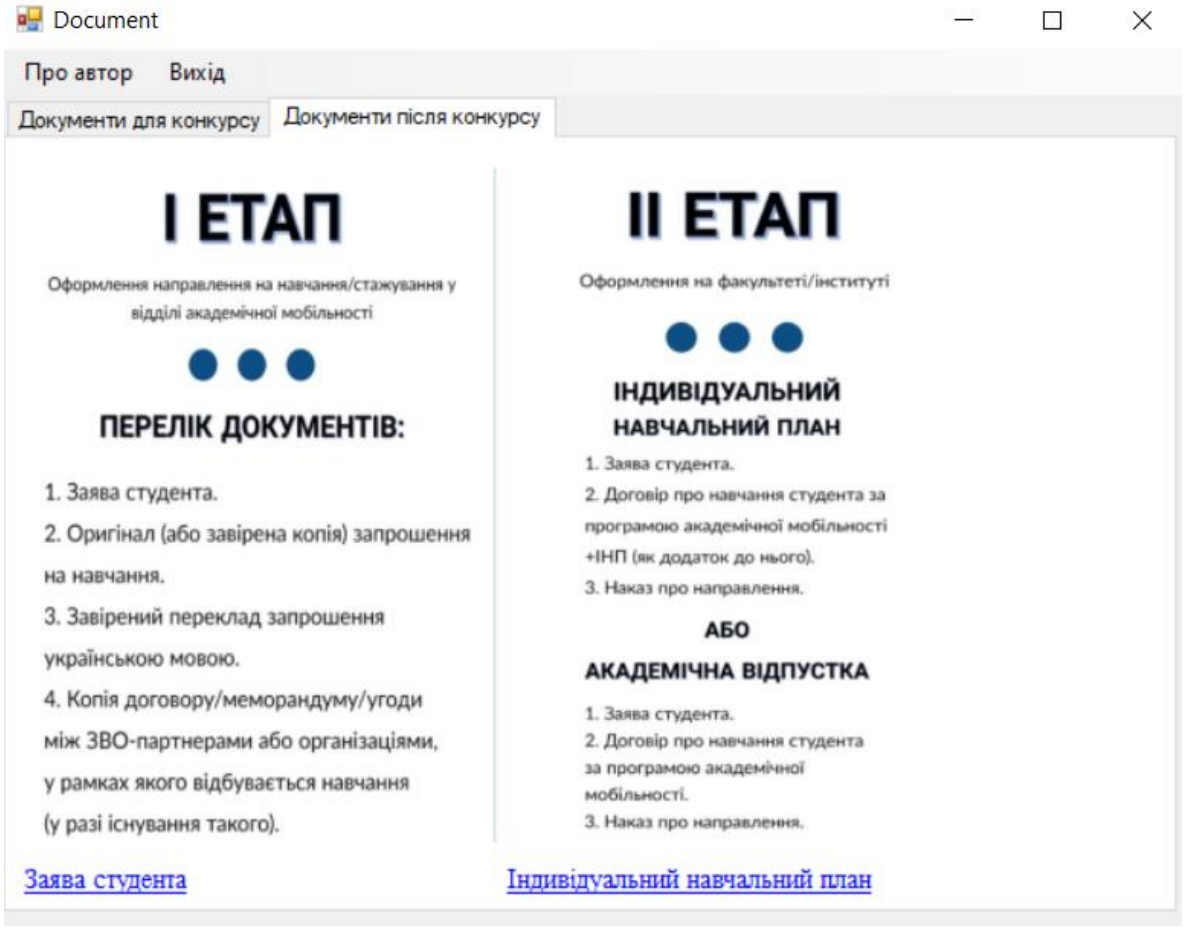


Рис.2.3—Основні документи необхідно подати після конкурсу[3]

На рисунку 2.4 показаний бланк заяви-анкети в паперовому вигляді

A. ВІДОМОСТІ ПРО КАНДИДАТА			
1	**ПІБ**		
	Серія та номер закордонного паспорту (із терміном закінчення)		
	Адреса електронної пошти, контактний номер телефону		
2	**Факультет/інститут**		
	Курс та група		
	Напрям підготовки (спеціальність та спеціалізація)		
	Рівень вищої освіти		
3	**Приймаючий заклад (за пріоритетом):**		
	- перший виш-партнер		
	- другий виш-партнер		
B. ДОДАТКОВА ІНФОРМАЦІЯ			
4	**Середній бал навчання**		**ПІБ та підпис координатора з академічної мобільності:**
5	**Чи брали Ви вже участь у проектах кредитної мобільності програми «Еразмус»?**	☐НІ	☐ ТАК _____ _____ _____ **(вказати кількість разів, які проекти)**
6	**Належність до пільгової категорії (інваліди тощо)*?** Якщо «так», зазначити до якої	☐ ТАК _____	☐НІ
7	**ЗНАННЯ ІНОЗЕМНОЇ МОВИ***		
	Рівень:	**Тип сертифікату:**	

Рис. 2.4 —Паперовий бланк заяви-анкети

На рисунку 2.5 показано паперову заяву для студентів при оформленні ІНП (індивідуального навчального плану)

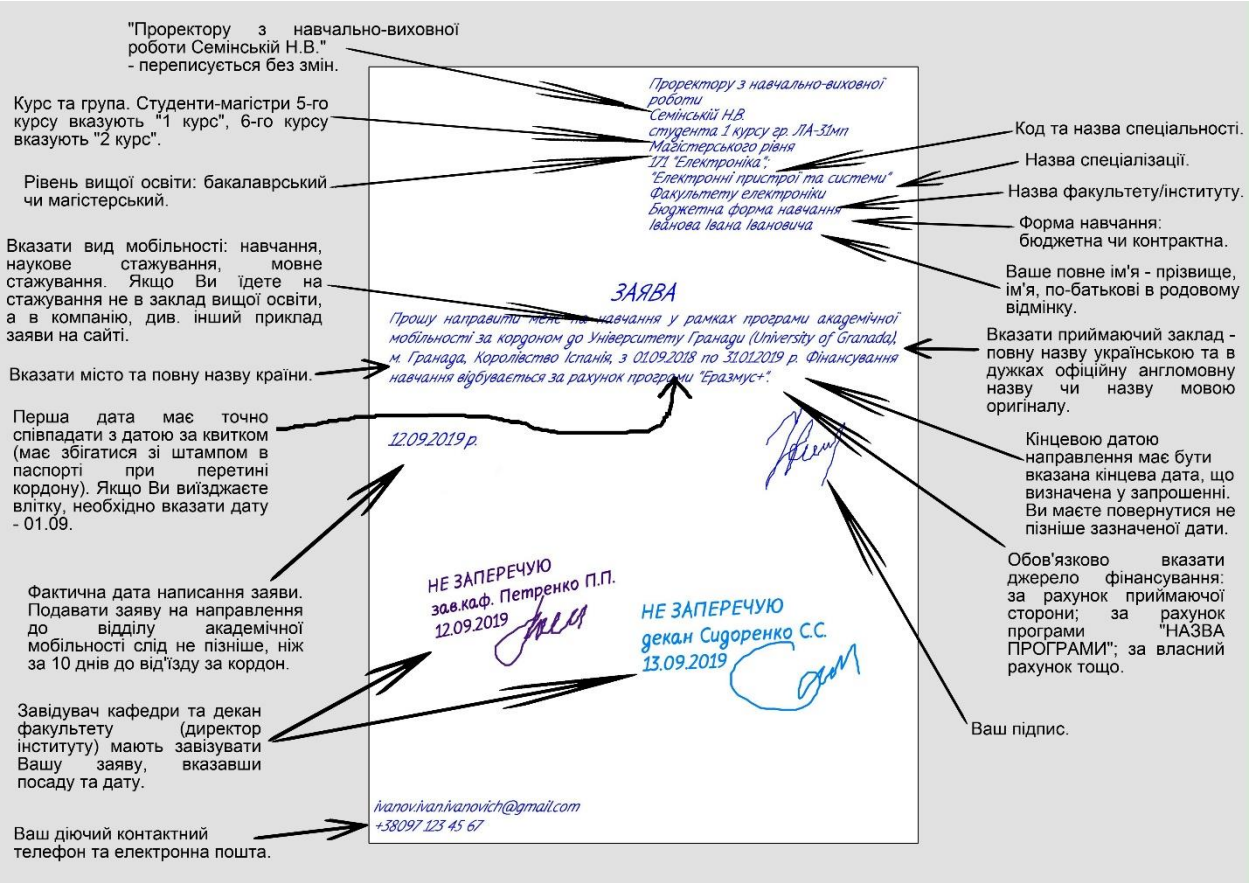


Рисунок 2.5 — Зразок заповнення заяви для студента

На рисунку 2.6 показано форма ІНП .

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

_____
(назва структурного підрозділу)

ЗАТВЕРДЖЕНО
Наказом декану
факультету/директором інституту

від «___»_____ _____ р. № ___

ІНДИВІДУАЛЬНИЙ НАВЧАЛЬНИЙ ПЛАН № ___
на період з _____ до _____ 20__/20__ навчального року

ПІБ студента _____
Назва факультету/інституту _____
Напрям підготовки/спеціальність, шифр групи _____ (шифр та назва)
ПІБ та посада координатора з академічної мобільності _____

№	Назва кредитного модуля	Кількість кредитів ЄКТС	Вид звітності	Дата семестрового контролю	Примітки
Осінній семестр					

Рис. 2.6: Індивідуальний навчальний план

Висновок до розділу

В розділі було показано вхідні дані для роботи з системою, розроблено бази даних та детально описано всі сутності в базі даних. А також, розглянуто всі необхідні документи для подачі на участь в конкурсі та зразок їхнього заповнення .

3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

В дипломному проекті розглянуто 2 основні задачі.

1) Обчислення рейтингу для окремого учасника.
2) Визначення фактору, який має найбільше вплив на результат конкурсу.

Задача 1 - «Обчислення рейтингу для окремого учасника».

Рейтинг учасника обчислюється сумою всіх вхідних параметрів. Для рейтингу необхідно розглядати такі основні параметри:

- середній бал учасника;
- рівень володіння мови;
- наявність публікації або статті ;
- перемога в олімпіадах.

Задача 2 - «Визначити, який фактор має найбільше вплив на результат конкурсу».

3.1. Змістовна постановка задачі

Нехай маємо N учасників, які беруть участь в конкурсі програми обміну. Вхідними даними для кожного учасника є такі фактори:

- середній бал навчання;
- середній бал по спеціальності;
- рівень володіння мовою;
- наявність участі в конференціях;
- наявність публікацій;
- наявність перемоги в олімпіадах.

Визначити, фактор, якій має найбільше вплив на результат рейтингу.

3.2. Математична постановка задачі

Дано:

Маємо N учасників конкурсу програм європейського обміну

X[i, j] – оцінка і-ого студента в j-ому факторі

Вихідні дані:

C[i,j] – кінцевий результат факторного аналізу

3.3. Обгрунтування методу розв’язання

Для розв’язання даної задачі, використуємо метод факторного аналізу[6]. Факторно-аналітичний підхід грунтується на уявлені про комплексний характер досліджуваного явища, що виявляється, зокрема, увзаємозв'язках між окремими його ознаками. Мета факторного аналізу – сконцентрувати вихідну інформацію, представлену у вигляді масиву даних і виразити якомога більшу кількість ознак через якомога меншу кількість характеристик [4]. Для нашої задачі маємо 6 факторів, які мають вплив на результат конкурсу - рейтинг. Найбільш впливовий на оцінку рейтинга фактор назвемо узагальненим фактором.

Узагальнені фактори конструюють таким чином, щоб максимально пояснити наявні кореляції між всіма факторами,які зазначені як вхідні дані.

Нехай висунута гіпотеза, що кожна з вихідних ознак $x^{(i)}$ лінійно залежить від загальних для всіх ознак факторів $y^{(j)}$, j=1–q, число яких *q* менше числа вихідних ознак *p*. Таке уявлення може бути лише приблизним, оскільки передбачається, що *p*-мірні дані лежать в *q*-мірному просторі. Щоб врахувати виникаючу через це помилку, вважають, що в кожній ознаці є деяка індивідуальна складова $e^{(i)}$, яка називається нев'язкою або специфічним (для цієї ознаки) фактором. Звичайно для зручності вважають, що дисперсії як загальних, так і специфічних факторів дорівнюють одиниці. Математично кожен фактор описується набором коефіцієнтів наступного вигляду:

$$X^{(1)} = b_{11}y^{(1)} + b_{12}y^{(2)} + \ldots + b_{1q}y^{(q)} + d_1e^{(1)}$$

$$X^{(2)} = b_{21}y^{(1)} + b_{22}y^{(2)} + \ldots + b_{2q}y^{(q)} + d_2e^{(2)}$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.1)$$

$$X^{(1)} = b_{p1}y^{(1)} + b_{p2}y^{(2)} + \ldots + b_{pq}y^{(q)} + d_pe^{(p)}$$

Матрицю $B = \{b_{ij}\}$ називають матрицею навантажень факторів на ознаки, або матрицею факторних навантажень.

Кожний її елемент показує вклад фактора $y^{(i)}$ в опис i-Ї ознаки, а коефіцієнт характеризує долю непоясненої мінливості $x^{(i)}$ при вибраному наборі загальних факторів. Суму квадратів навантажень на будь-якому рядку матриці B називають спільністю відповідної ознаки; чим більше це значення, тим краще описується ознака загальними факторами. Спільність є частиною дисперсії ознаки, яку «пояснюють» загальні фактори. Квадрат коефіцієнта при специфічному факторі називають специфічністю ознаки. Ця величина показує, яка частина дисперсії вихідної ознаки залишається «непоясненою» при даному наборі загальних факторів, інакше кажучи,наскільки ознака не залежить від загальних факторів. Таким чином, дисперсія ознаки $D(X^{(i)})$ = спільність $\sum_{i=1}^{p}(d_i^2)$ + специфічність (d_i^2)

Факторний аналіз починають з пошуку факторних навантажень, щоб визначити число факторів q. Процедура носить ітераційний характер: починають працювати з $q = 1$ (однофакторна модель); отримавши навантаження, перевіряють, наскільки кореляційна матриця, відновлена по однофакторній моделі, відрізняється від кореляційної матриці вихідних даних. Якщо ця відмінність незначна, то модель визнається задовільною (адекватною), у протилежному випадку випробовується модель із $q = 2$ і т. д.

доти, доки при деякому q не буде досягнута адекватність або число факторів у моделі не перевищить максимально припустиме (визначаються співвідношенням$(q<(p-1)/2)$, інакше число невідомих величин у моделі стає більше числа рівнів, за якими їх можна визначити. У останньому випадку говорять, що адекватної моделі факторного аналізу не існує.

На цьому рішення задачі факторного аналізу не закінчується. Якщо адекватна модель існує, то необхідно (як у задачі аналізу головних компонент) виразити нові ознаки (знайдені з загальних факторів) через старі і побудувати нову таблицю із $q<p$ ознаками. Тільки після цього обчислюють навантаження ознак на фактори, тобто виражають фактори (нові ознаки) через старі:

$$Y^{(1)} = C_{11}x^{(1)} + C_{12}x^{(2)} + \ldots + C_{1p}x^{(p)}$$

$$Y_{(2)} = C_{q1}x^{(1)} + C_{q2}x^{(2)} + \ldots + C_{qp}x^{(p)}$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (3.2)$$

$$Y_{(q)} = C_{q1}x^{(1)} + C_{q2}x^{(2)} + \ldots + C_{qp}x^{(p)}$$

Матриця $C = \{C_{ij}\}$ і є кінцевим результатом факторного аналізу, який представляє нову таблицю з q незалежними ознаками

3.4. Опис методів розв'язання

Із даними учасників конкурса створюємо таблицю з розміром N*6, де N – це кількість учасників конкурсу і 6 факторів, які описують учасників.

Створюємо таблицю, в якій середній бал задаємо в 100-бальній шкалі, рівень володіння мови задаємо в шкалі від 1 до 4, де оцінка 1 - це означає рівень іноземної мови учасника B1, якщо оцінка дорівнює 4 - це означає рівень мови C2.

Так же робиться з наявністю публікацій та наявністю перемоги в конкурсі. Оцінка 1 відповідає за відсутність публікації та конкурсу, а оцінка 4 означає, що учасник брав участь у конкурсі міжнародного рівня.

Пронормуюмо таблицю даних у 10-бальну шкалу. Так як для участі в конкурсі учасник повинен мати середній бал не нижче ніж 75 балів, вважаємо 75 балів – 1 бал в нормованій таблиці, а 100 балів – 10 балів. В результаті, отримаємо таку формулу:

$$\text{Нормований ср. бал} = 1 + (\text{ср. бал} - 75) * \frac{9}{25} \qquad (3.3)$$

Аналогічна формула для нормування середнього балу по спеціальності.

Для рівня володіння мови, наявності публікацій та перемоги в олімпіадах, маємо формулу: $\text{нормована оцінка} = \text{оцінка} * 2.5$

Отже, отримаємо таку таблицю де найнижча оцінка для факторів дорівнює 1, а найвища оцінка – 10 (рисунок 3.1)

	1 ср. бал	2 Сп.ср.бал	3 Англ	4 Конф	5 публ	6 конк
1	5,5	4,348	2,5	2,5	2,5	10
2	1,216	1,936	5	2,5	2,5	10
3	6,112	2,566	5	10	5	5
4	1,648	1,36	7,5	2,5	2,5	5
5	6,256	6,4	2,5	10	10	5
6	3,664	1,9216	7,5	2,5	2,5	10
7	3,628	2,836	5	2,5	5	10
8	5,14	4,996	2,5	5	10	10
9	4,78	5,212	2,5	2,5	5	10
10	5,284	1,324	2,5	2,5	10	10
11	7,264	5,842	5	5	5	10
12	8,632	5,536	7,5	7,5	5	5
13	5,5	2,008	5	5	10	5
14	5,122	5,104	7,5	2,5	5	5
15	2,314	5,104	7,5	2,5	2,5	5
16	1,666	1,18	7,5	2,5	2,5	2,5
17	5,068	2,26	2,5	5	2,5	2,5
18	5,644	6,4	2,5	2,5	5	2,5
19	4,96	5,428	2,5	5	2,5	2,5
20	4,472	5,032	2,5	5	2,5	2,5
21	6,94	5,788	7,5	10	5	2,5
22	4,348	3,88	7,5	2,5	5	2,5
23	3,5776	4,24	5	10	2,5	2,5
24	7,336	6,454	5	5	5	10
25	8,416	7,2172	2,5	5	10	10
26	5,144	6,1	5	7,5	2,5	5

Рисунок 3.1 —Таблиця вихідних даних (фрагмент)

Стандартизуємо всі вихідні дані для проведення аналізу в системі STATISTICA і використовуємо модуль «Факторний аналіз»[6].

Одержуємо кореляційну матрицю вихідних даних для подальшого аналіза (рисунок 3.2)

Correlations (Spreadsheet1) Marked correlations are significant at p < ,05000 N=50 (Casewise deletion of missing data)

	ср. бал	Сп.ср.бал	Англ	Конф	публ	конк
ср. бал	1,000000	0,753839	-0,166032	0,334366	0,405551	0,148433
Сп.ср.бал	0,753839	1,000000	-0,156340	0,352973	0,151792	-0,031526
Англ	-0,166032	-0,156340	1,000000	0,025642	-0,139839	-0,237827
Конф	0,334366	0,352973	0,025642	1,000000	0,080024	-0,176405
публ	0,405551	0,151792	-0,139839	0,080024	1,000000	0,237324
конк	0,148433	-0,031526	-0,237827	-0,176405	0,237324	1,000000

Рисунок 3.2 —Кореляційна матриця даних

Для виділення узагальненого фактора отримаємо власні значення факторів (рисунок 3.3), які описують внесок в загальну дисперсію кожого з шести факторів

	Factor Loadings (Unrotated) (Spreadsheet1) Extraction: Principal components (Marked loadings are >,700000)	
Variable	Factor 1	Factor 2
ср. бал	-0,913181	-0,035618
Сп.ср.бал	-0,830634	-0,257073
Англ	0,324091	-0,517228
Конф	-0,502689	-0,548623
публ	-0,525253	0,396578
конк	-0,186548	0,782171
Expl.Var	2,192275	1,404932
Prp.Totl	0,365379	0,234155

Рисунок 3.3 —Власні значення факторів

Отримано інформацію про те, скільки дисперсії виділив кожен фактор, тому можемо виявити скільки чинників слід залишити. Використовуємо критерій кам'янистій осипи: - критерій кам'янистій осипи є графічним методому де спадання власних значень зліва направо максимально передбачає, що праворуч від цієї точки знаходиться лише "факторіальнаосип" - "осип" є геологічним терміном, що позначає уламки гірських порід, що скупчуються в нижній частині скелястого схилу (рисунок 3.4). Як видно можно залишити два узагальнених фактора.

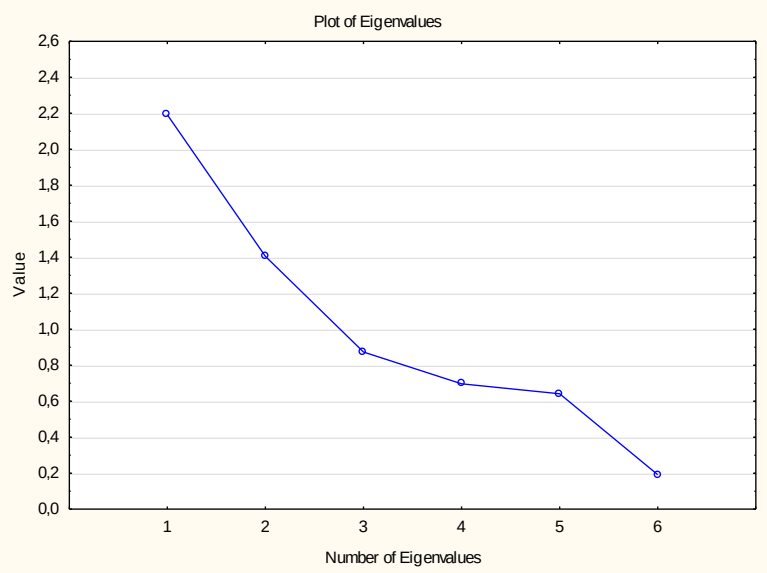


Рисунок 3.4— Критерій кам'янистій осипи

Для більшего розуміння результатів використаемо метод обертання варімакс (рисунок 3.5)

| | Factor Loadings (Varimax normalized) (Spreadsheet1
Extraction: Principal components
(Marked loadings are >,700000) | |
|---|---|---|
| Variable | Factor 1 | Factor 2 |
| ср. бал | 0,837095 | 0,366661 |
| Сп.ср.бал | 0,859521 | 0,131389 |
| Англ | -0,065741 | -0,606826 |
| Конф | 0,691779 | -0,274089 |
| публ | 0,299394 | 0,586113 |
| конк | -0,173677 | 0,785129 |
| Expl.Var | 2,042184 | 1,555023 |
| Prp.Totl | 0,340364 | 0,259170 |

Рисунок 3.5 — Метод отримання зрозумілої матриці навантажень

Маємо зрозумілу інтерпритацію, що два фактори можуть замінити оцінювання по шесті факторах – це узагальнена оцінка по середньому балу і середньому балу по спеціальним дисциплінам і участь в олімпіадах (рисунок 3.7).

| | Eigenvalues (Spreadsheet1)
Extraction: Principal components | | | |
|---|---|---|---|---|
| Value | Eigenvalue | % Total variance | Cumulative Eigenvalue | Cumulative % |
| 1 | 2,192275 | 36,53791 | 2,192275 | 36,53791 |
| 2 | 1,404932 | 23,41553 | 3,597207 | 59,95344 |

Рисунок 3.6 — Факторні навантаження

Факторні навантаження можна інтерпретувати як кореляції між узагальненими факторами і вхідними факторами.Найбільше навантаження має узагальнений фактор 1, який пояснює взаємовплив всіх 6-ти факторів на рейтинг .

Для остаточного висновку проведемо регресійний аналіз з виділеними факторами.

Regression Summary for Dependent Variable: ср. бал (Spreadsheet1)
R=,753839 R?=,568273 Adjusted R?=,559279
F(1,48)=63,181 p<,00000 Std.Error of estimate:1,2854

N=50	Beta	St. Err. of Beta	B	St. Err. of B	t(48)	p-value
Intercpt			1,371517	0,498715	2,750103	0,008376
Сп.ср.бал	0,753839	0,094838	0,765206	0,096268	7,948673	0,000000

Рисунок 3.7 —Результати регресійного аналіза

Результати аналіза показують, що середній бал і середній бал з спеціальних дисциплін лінійно залежні і можно обраховувати значення середнього балу як рейтінг, який враховую взаємодію всіх шесті факторів, т.т. рейтинг, обрахований за методом факторного аналіза відображує узагальнений вплив всіх 6-ти факторів, які характеризують учасника конкурсу.

Висновок до розділу

В цьому розділі, було розглянуто основні задачі, які небхідно розв’язати в цьому проекті, це обчислення рейтингу для кожного учасника та визначення узагальненого фактору, який має найбільший вплив на результати конкурсу і може бути використаний як оцінка учасника. Викладено обґрунтування метода розв’язання і проведено аналітичні розрахунки для підтвердження результату..

4. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Засоби розробки

Для розробки даного проекту, було використано мову програмування - C#, технологія Window Forms, і бази даних – SQL Server Management 2017. Вибір мова програмування C# для розробки, тому що це об'єкт-орієнтована мова програмування. C# відноситься до відомого сімейство мови C, з синтаксисом схоже на синтаксису C++ та Java[7,8]. C# був засновано на основі мови програмування Java та C++, звідси синтаксис C# є простим і близьки до C++ та Java. Для розробки факторного аналізу, маємо бібліотека C#, який підтримує цей метод, бібліотека Extreme.Data, Extreme.Statistics,... Вибір Window Forms як технологія розробки проекту, в першу чергу тому що він є простим для розробки інтерфейсу. Для програмування, мій вибір впав на Visual Studio 2017, оскільки це є зручним засобом для роботи з Window Forms та бази баних SQL Server 2017.

4.2. Вимоги до технічного забезпечення

4.2.1. Загальні вимоги

Мінімальні вимоги до технічного забезпечення:

- операційна система – Window 7 з пакетом оновлення SP 1 [8];
- оперативна пам'ять – 2 ГБ [8];
- місткість жорсткого диску – до 50 ГБ;
- процессор – 1.8 ГГц [8].

Рекомендовані вимоги до технічного забезпечення:

- операційна система – з Window Server 2012 R2 [8];
- оперативна пам'ять – 4 ГБ [8];
- місткість жорсткого диску – більше 50 ГБ;
- процессор – більше 1.8 ГГц [8].

4.3. Архітектура програмного забезпечення

4.3.1. Діаграма класів

На рисунку 4.1 показує діаграма класів серверної частини чата

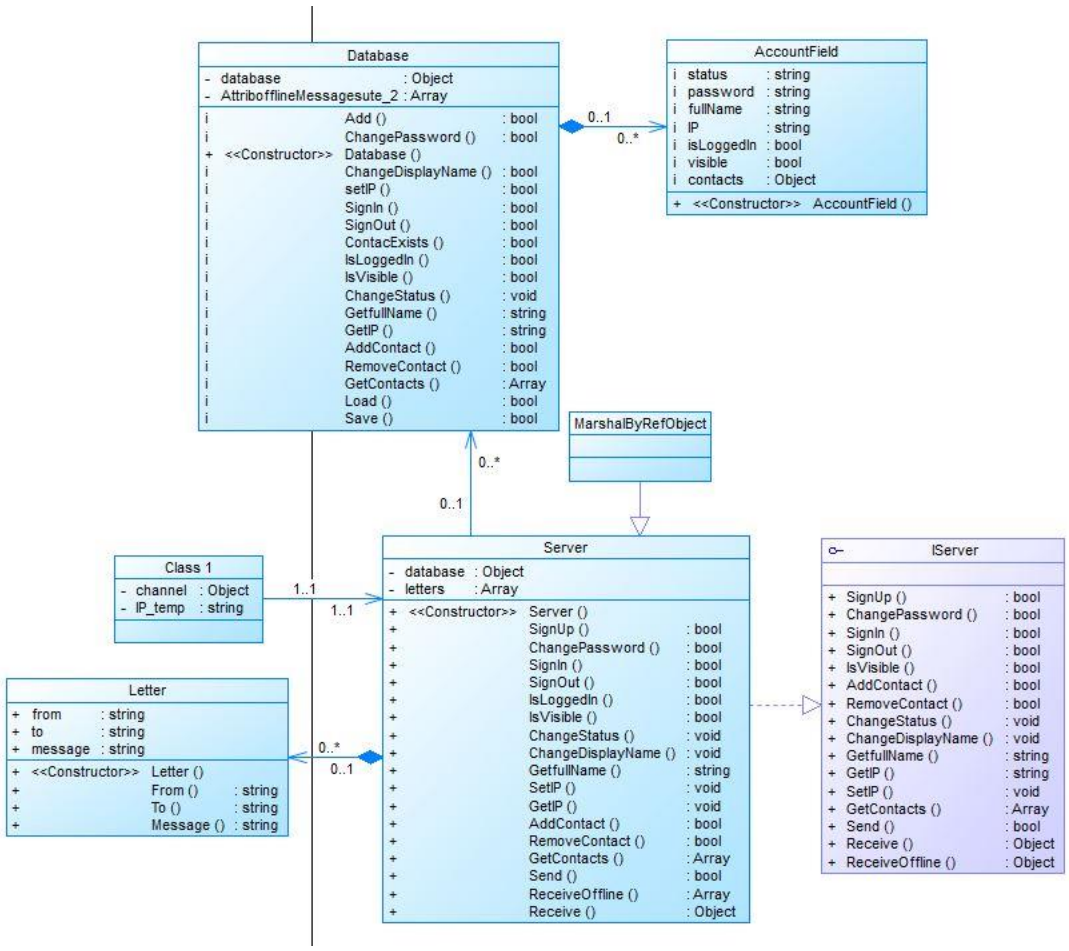


Рисунок 4.1 — Діаграма класів серверної частини

4.3.2. Діаграма послідовності

На рисунку 4.2 показана діаграма послідовності для актора «Студент»

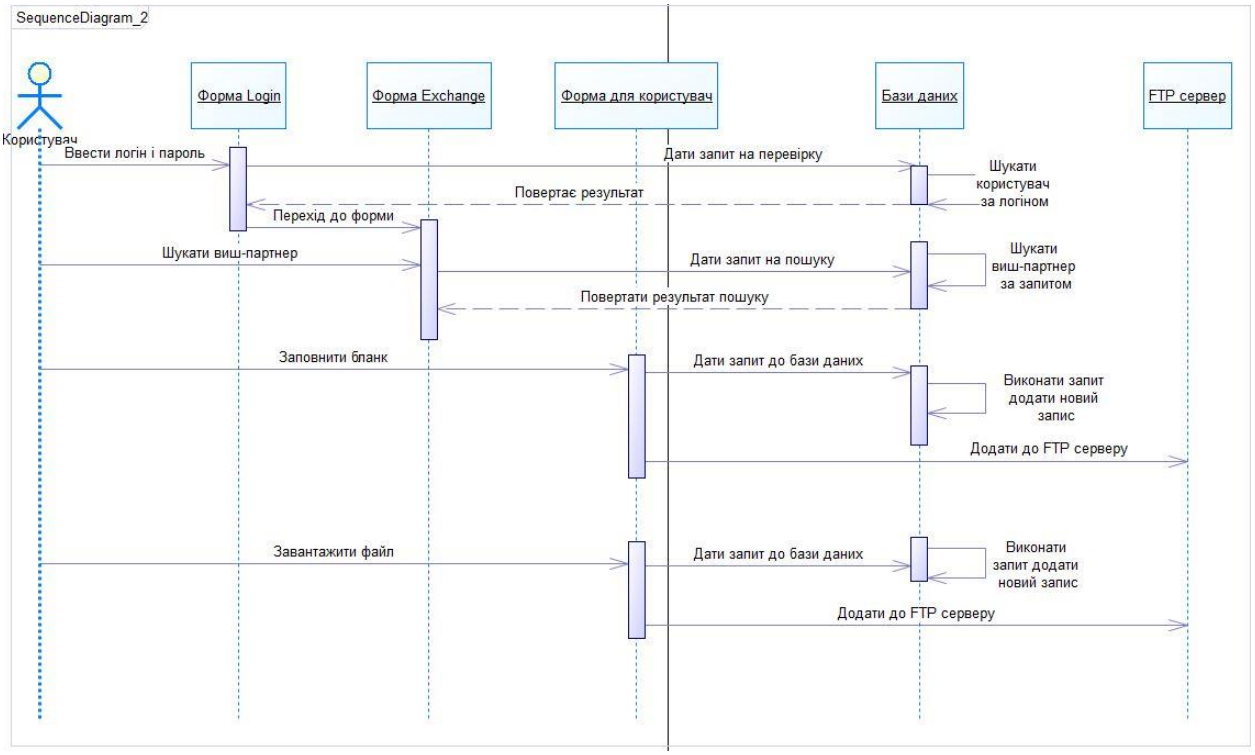


Рисунок 4.2 —Діаграма послідовності для актора «Студент»

4.3.3. Діаграма компонентів

На рисунку 4.3 показує діаграми компонентів системи

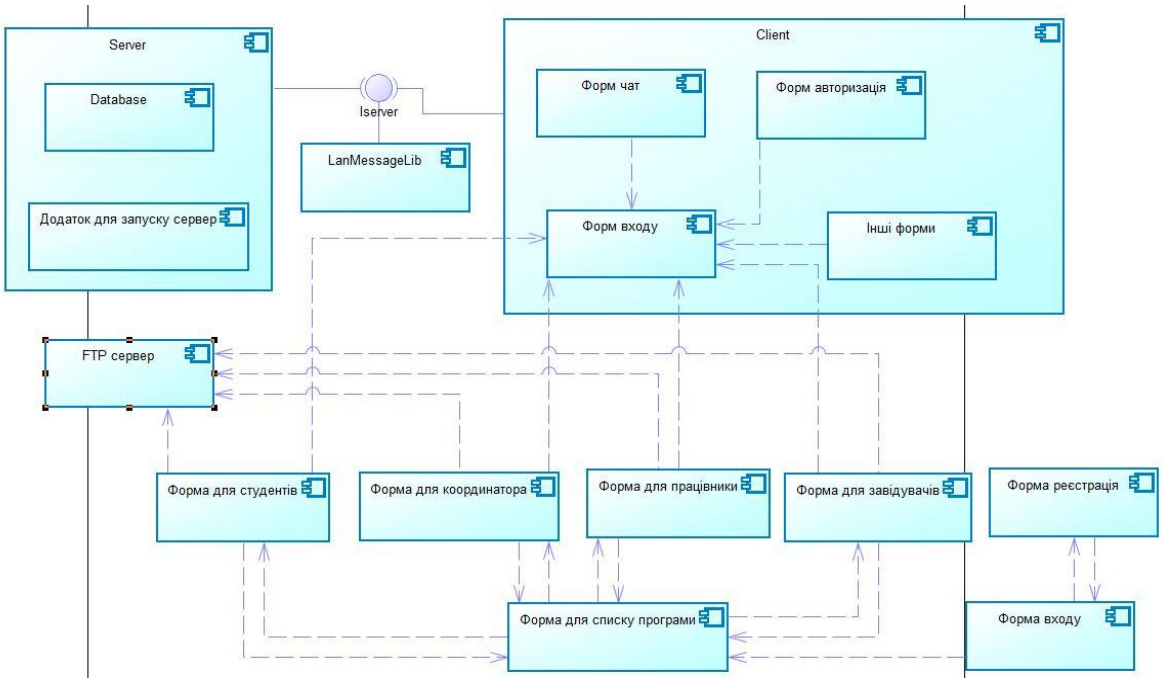


Рисунок 4.3 — Діаграма компонентів

4.3.4. Специфікація функцій

В таблиці 4.1 наведено основні функції, які мають головну роль в системі

Таблиця 4.1 – Таблиця основних функцій

Назва класу	Назва функції	Призначення
Login : Form	CheckLogin(stringusername, stringpassword)	Перевірка існуючого аккаунту. Вхід в системі
Registration : Form	public void regis()	Додати новий аккаунт до системи.
Exchange : Form	public void Search_Univer ()	Шукати виш-партнер для факультету користувача
User_Form : Form	public void apply_document()	Подати документи для участі в конкурсі
User_Form : Form	public void UploadFile()	Відправити файл документів для участі в конкурсі до FTPсерверу
User_Form : Form	public void Apply_1_etap()	Подати документи першого етапу після конкурсу до FTPсервера
User_Form : Form	public void UploadFile_1()	Відправити файл документів першого етапу після конкурсу до FTPсервера
User_Form : Form	public void UploadFile_2()	Відправити файл документів другого етапу після конкурсу до FTPсервера
User_Form : Form	public void Apply_2_etap()	Подати документи першого етап після конкурсу до FTPсервера

Продовження таблиці 4.1

Назва класу	Назва функцій	Призначення
Database	internal bool ChangePassword(string username, string curPassword, string newPassword)	Змінити пароль
Database	internal bool Add(string username, string password, string fullName, string IP)	Додати нового користувача
Database	internal bool SetIP(string username, string IP)	Перевірка даного аккаунту для встановлення нового IP-адреса
Database	internal bool SignIn(string username, string password, bool visible)	Перевірка успішності входу до системи
Database	internal bool SignOut(string username)	Перевірка успішності виходу з системи
Database	internal bool ContactExists(string username)	Перевірка існуючого аккаунту
Database	internal bool IsLoggedIn(string username)	Перевірка на авторизацію
Database	internal void ChangeStatus(string username)	Змінити статус
Database	internal string GetfullName(string username)	Отримати повне ім’я
Database	internal string GetIP(string username)	Отримати IP-адрес
Database	internal bool AddContact(string username, string contact)	Перевірка даного аккаунту для додавання до списку контактів
Database	internal bool RemoveContact(string username, string contact)	Перевірка даного аккаунту для видалення із списку контактів
Database	internal ArrayList GetContacts(string username)	Отримати повний список контактів
Database	internal bool Load()	Перевірка успішності завантаження
Server	public bool SignUp(string username, string password, string fullName, string IP)	Реєстрація

Продовження таблиці 4.1

Назва класу	Назва функцій	Призначення
Database	internal bool Save()	Перевірка успішності зберігання
Server	public bool SignIn(string username, string password, bool visible)	Перевірка успішності входу до системи
Server	public bool SignOut(string username)	Перевірка успішності виходу із системи
Server	public bool IsLoggedIn(string username)	Перевірка на авторизацію
Server	public void ChangeStatus(string username)	Змінити статус
Server	public string GetfullName(string username)	Отримати повне ім’я
Server	public void SetIP(string username, string IP)	Встановити IP-адрес
Server	public string GetIP(string username)	Отримати IP-адрес
Server	public bool AddContact(string username, string contact)	Перевірка даного аккаунту для додавання до списку контактів
Server	public bool RemoveContact(string username, string contact)	Перевірка даного аккаунту для видалення із списку контактів
Server	public ArrayList GetContacts(string username)	Отримати повний список контактів
Server	public bool Send(string from, string to, string message)	Перевірка успішності відправлення повідомлення
Server	public ArrayList ReceiveOffline(string to)	Список затриманих повідомлень
Server	public LetterReceive Receive(string to)	Отримати повідомлення
Employee : Form	public void Download(string folder, string fileName)	Завантажити документи
Employee : Form	public void Scoring()	Дати оцінку за публікацію і перемоги в конкурсах

Продовження таблиці 4.1

Назва класу	Назва функції	Призначення
SampleObject	public void SendMsgToSvr(string chatMsgFromUsr)	Відправити повідомлення до сервера
SampleObject	public string GetMsgFromSvr(int lastKey)	Отримати повідомлення із сервера
Server	public bool ChangePassword(string username, string curPassword, string newPassword)	Змінити пароль
SampleObject	public bool JoinToChatRoom(string name)	Перевірка успішності приєднання до групового чату
SampleObject	public void LeaveChatRoom(string name)	Вийти з групового чату
SampleObject	public ArrayList GetOnlineUser()	Отримати список онлайн-користувачів

Висновок до розділу

В цьому розділі, було наведено діаграма класів, діаграма послідовності, діаграма компонентів. За допомогою цих діаграм, ми можемо розуміти послідовність дії для акторів в системі та структуру програми. Крім цього, було наведено список основних функцій, де основне призначення цих функцій та в якому класі вони знаходяться.

5. ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1. Керівництво користувача

Для початку роботи, система відкриє для користувача вікно для авторизації (форма Login), якє показано на рисунку 5.1, і для початку роботи з системою користувачу необхідно ввести свій логін та пароль.

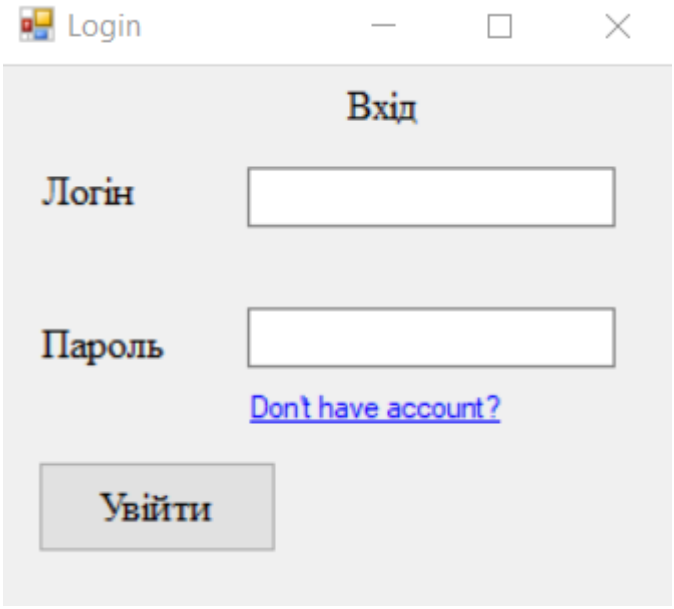


Рисунок 5.1 — Форма авторизації

Якщо немає аккаунту, натиснути посилання «Don’thaveaccount?», система відразу відкриє для вас вікно для реєстрації (форма Registration). Для реєстрації, користувача неообхідно ввести такі дані як ПІБ, логін, назва факультету, кафедри, рік навчання, ваша позиція в роботу, пароль.Всі поля повинен бути заповненим для подальшої роботи. На рисунку 5.2 показана форма для реєстрації.

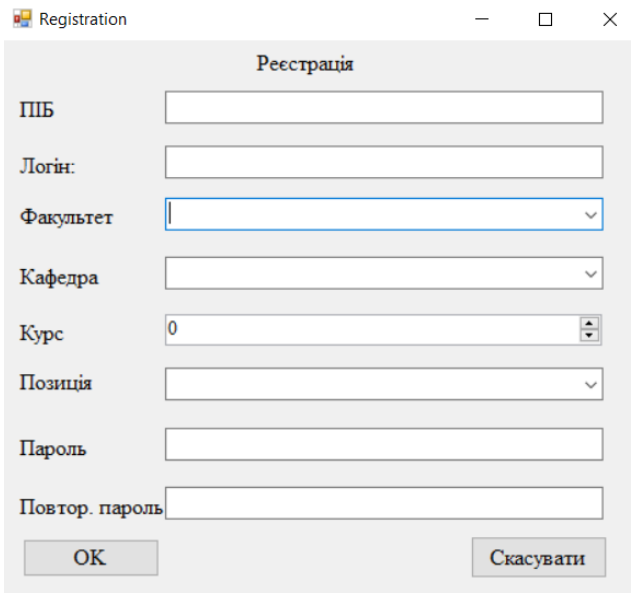


Рисунок 5.2 — Форма реєстрації

Якщо користувач правильно ввів логін і пароль, система відкриє форму для програми європейських обмінів. Після відкриття формив DataGridView показується всі програми, які існують. А також, можно вибирати виш-партнера з випадаючого списку, щоб почитати інформації про цей університет. Крім цього, натиснувшиCheckBox, в формі показує всі виш-партнери, які мають контракт з вамим факультетом. На рисунку 5.3 і 5.4 наведено форма для програм обмінів

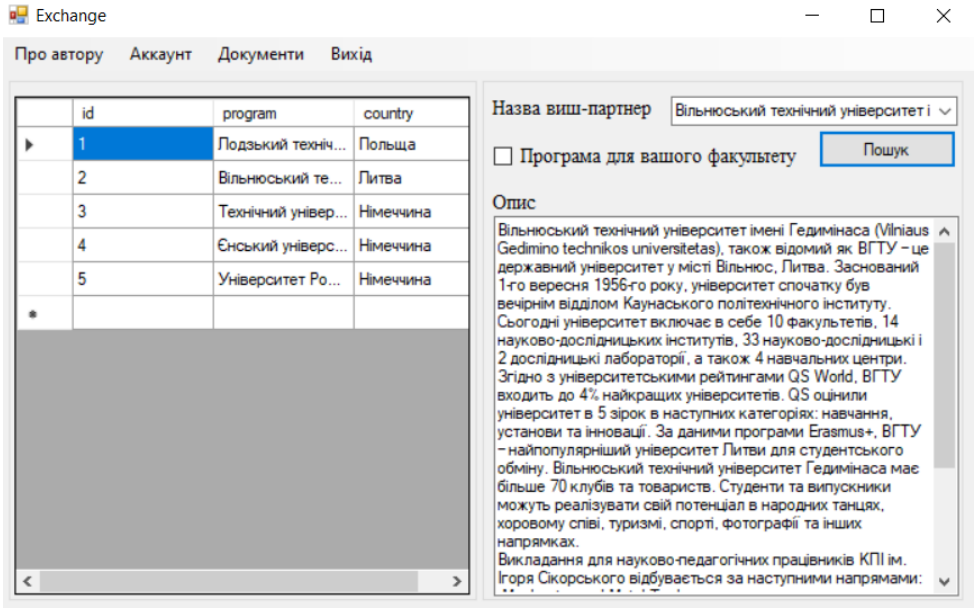


Рисунок 5.3 — Форма для програм обмінів

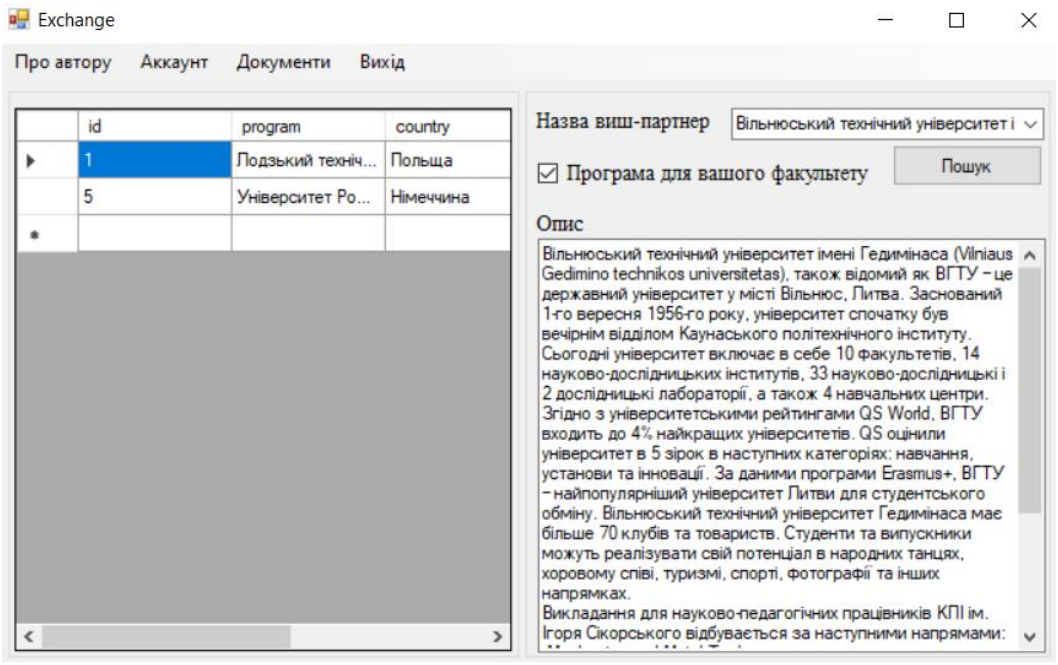


Рисунок 5.4— Форма для програм обмінів показує всі університети, які мають контракт з вашим факультетом

Для подальшої роботи в меню було показано 4 пункти: про автора, аккаунт, документи, вихід. Для пункту «Документи» після натиску система відкриває форму необхідних документів, де користувач може побачити всі необхідні документи для подачі та завантажити ці документи для заповнення. На рисунку 5.5 показано форму необхідних документів.

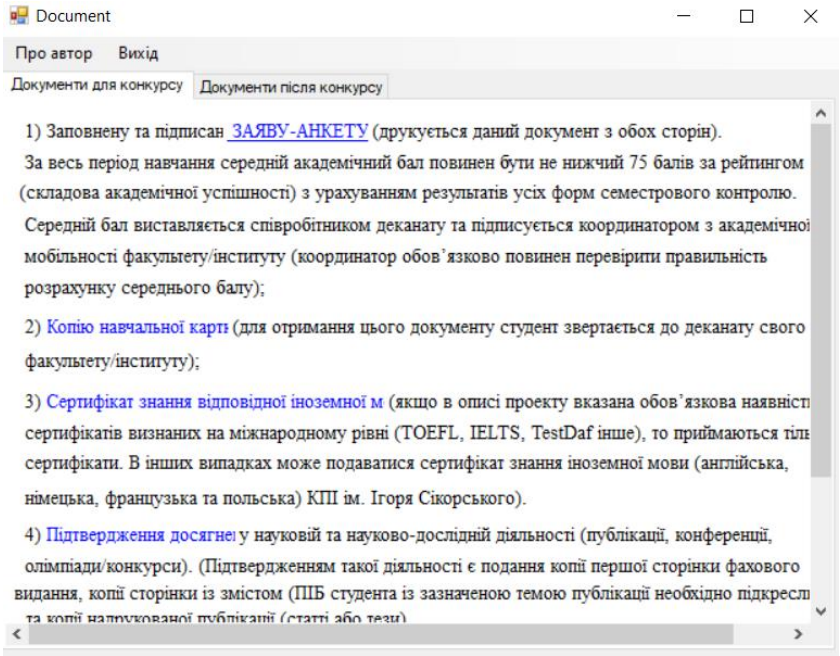


Рисунок 5.5— Форма необхідних документів

Після натискання пункту «Аккаунт», залежно від позиції користувача, система відкриває окрему форму для учасників, працівника відділу, адміністрації. На рисунку 5.5, 5.6,5.7 показано форми для учасників.

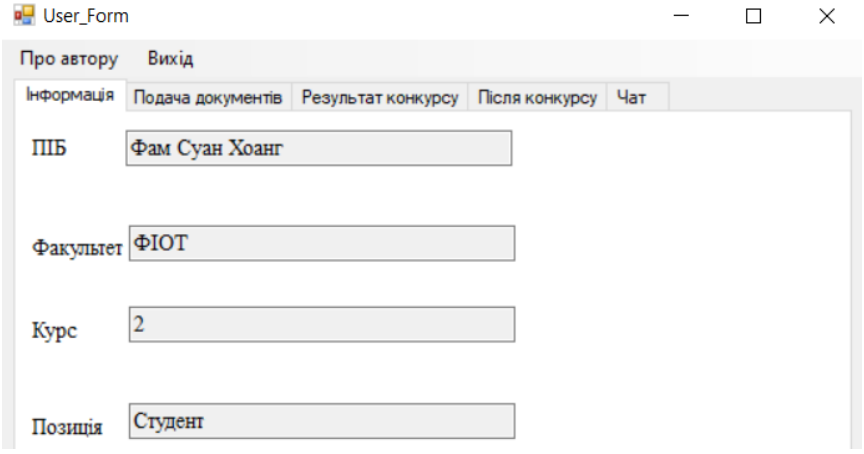


Рисунок 5.5— Форма з інформацією про учасника

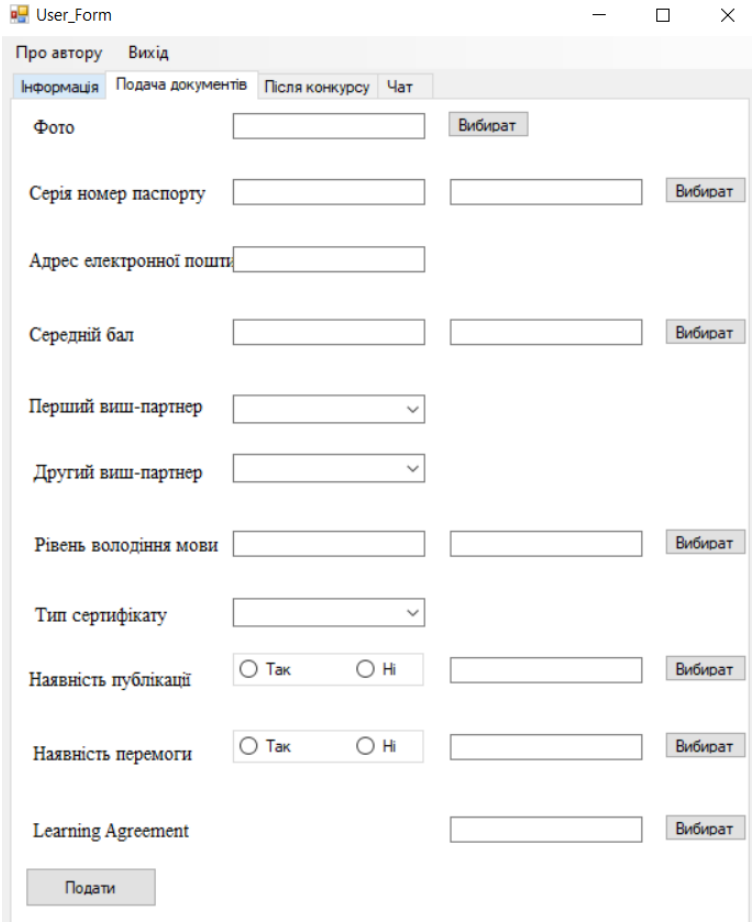


Рисунок 5.6— Форма для подачі документів для участі в конкурсі в електронному вигляді

На формі для працівника відділу, відображєний список учасників, список переможців, додатковий бал за публікації та перемоги в олмпіадах.

На рисунку 5.7, 5.8 показано список учасників конкурсу та вікно для оцінки за досягнення.

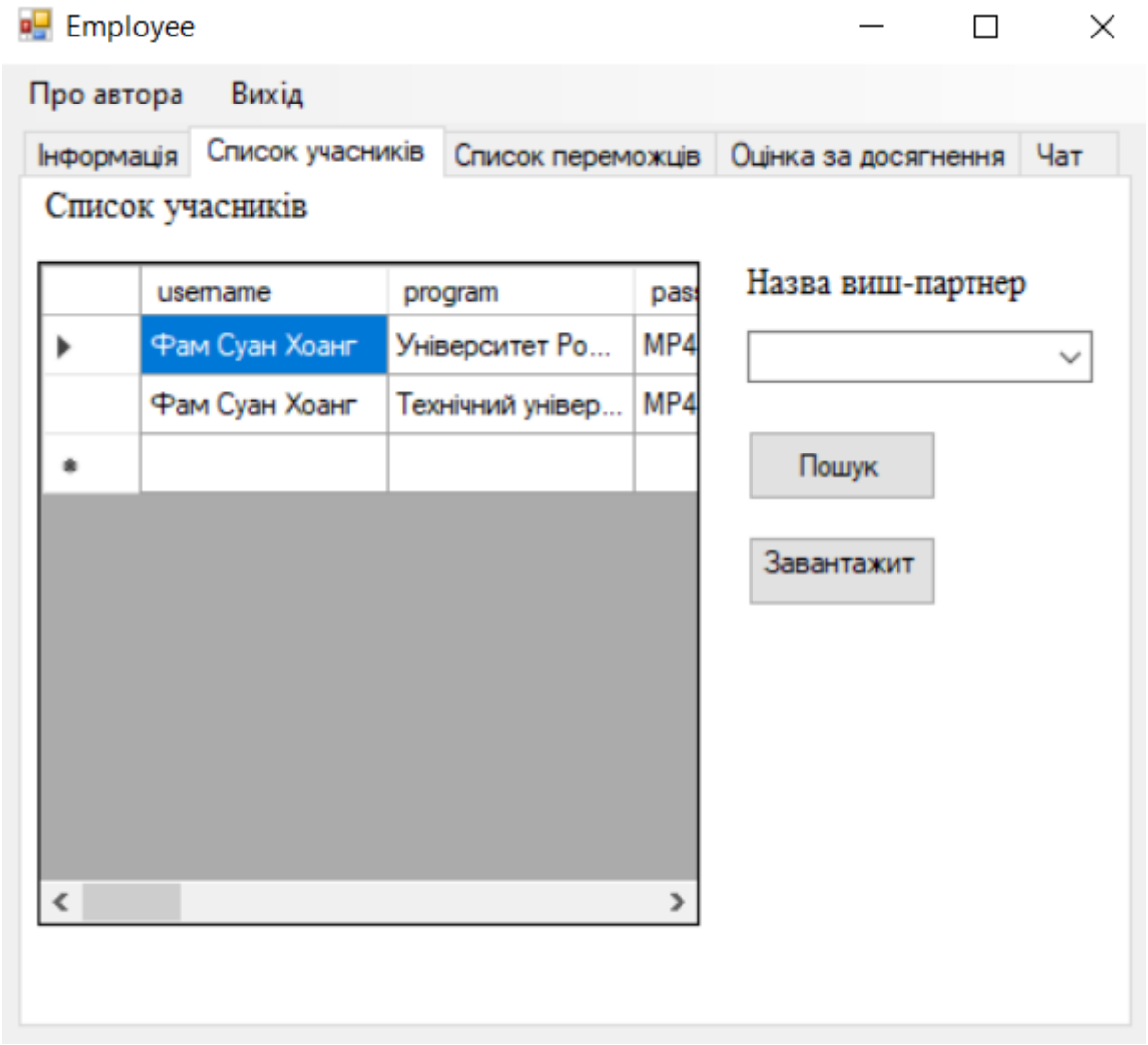


Рисунок 5.4— Форма показує список заявок на конкурс

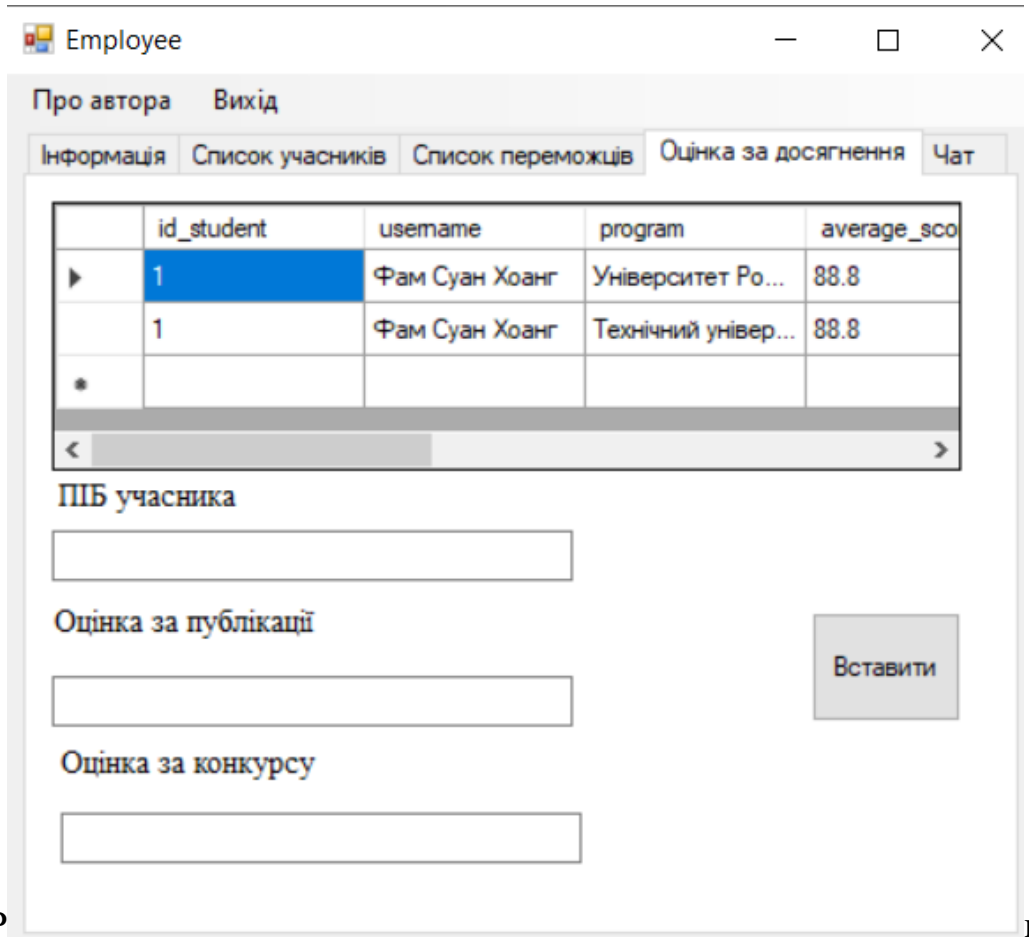


Рисунок 5.5— Форма для поставлення оцінки за досягнення

На рисунку 5.7 показано результат конкурсу

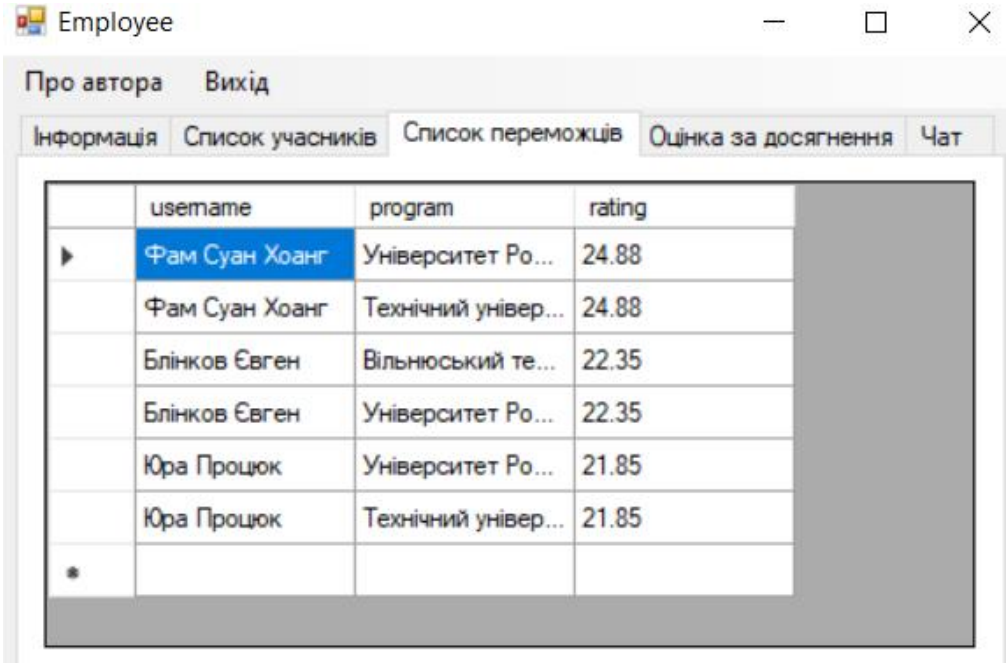


Рисунок 5.7—Результат конкурсу

5.2. Випробування програмного продукту

5.2.1. Мета випробувань

Мета випробування перевірка успішності подачі документів до конкурсу та після конкурсу. Працівники відділу мобільності зможуть отримати документи в електронному вигляді, обчислити рейтингучасників в конкурсі, а також поставити оцінки за досягнення для кожного учасника.

5.2.2. Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;

– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3. Результат випробувань

Тестування було на перевірку функціональності системи. Нижче наведено таблиці випробувань функціональності системи

Таблиця 5.1 – Перевірка випробування авторизації

Назва випробування	Випробування авторизації
Стан системи перед випробуванням	Система відкриває форму логіна
Вхідні дані	Логін і пароль системи
Вихідні результати	Якщо неправильно ввів логін і пароль, система показує повідомлення щодо неправильного вводу. Якщо правильно ввів логін і пароль, переходить в інші формі
Стан системи після випробування	Система відкриває форму для програми європейьких обмінів

Таблиця 5.2 – Перевірка випробування реєстрації

Назва випробування	Випробування реєстрації
Стан системи перед випробуванням	Система відкриває форму реєстрації
Вхідні дані	ПІБ, логін, пароль, назва факультету, кафедри, курс, позиція
Вихідні результати	Якщо існує пусте поле, система показує повідомлення про пусті поля. Якщо всі поля заповнені, реєтрація успішно проходить
Стан системи після випробування	Повертається до форми Логіна

Таблиця 5.3 – Перевірка випробування подачі документи до конкурсу

Назва випробування	Випробування подачі документи до конкурсу
Стан системи перед випробуванням	Система відкриває форму учасника
Вхідні дані	Фото, серії номер паспорту, адрес електронної пошти, середній бал, назва першого, другого виш-партнеру, рівень володіння мови, файл фото, скан паспорту, скан середнього балу, файл публікацій, файл перемоги конкурсу, файл плану навчання
Вихідні результати	Якщо існує пусте поле, система показує повідомлення про пусті поля. Якщо всі поля заповнені, всі дані зберігатимуться в базі даних
Стан системи після випробувань	Файл документів будуть зберігатися в FTPсервері

Таблиця 5.4 – Перевірка випробування подачі документи після конкурсу

Назва випробування	Випробування подачі документи після конкурсу
Стан системи перед випробуванням	Система відкриває форму учасника
Вхідні дані	Заява, запрошення, ІНП, контракт...
Вихідні результати	Якщо існує пусте поле, система показує повідомлення про пусті поля. Якщо всі поля заповнені, всі дані зберігатимуться в базі даних
Стан системи після випробування	Файл документів будуть зберігатися в FTPсервері

Таблиця 5.5 – Перевірка випробування поставлення оцінки та обчислення рейнтингу

Назва випробування	Випробування поставлення оцінки та обчислення рейнтингу
Стан системи перед випробуванням	Система відкриває форму для працівника відділу
Вхідні дані	Бал за конкурси, бал за публікації, ім’я користувачів
Вихідні результати	Якщо неправильно ввів ім’я учасника, система показує повідомлення про некоректне введення даних. Якщо всі дані правильно введені, система відразу обчислює рейтинг для окремого учасника
Стан системи після випробування	Додати нові зміни до бази данихі

Таблиця 5.6 – Перевірка випробування пошуку виш-партнер по факультету

Назва випробування	Випробування пошуку виш-партнер по факультету
Стан системи перед випробуванням	Система відкриває форму для програми обмінів
Вхідні дані	Назва факультету
Вихідні результатии	Список програми обмінів, яких відповідають вимогам
Стан системи після випробування	-

Таблиця 5.7 – Перевірка випробування завантаження файлів для підписування

Назва випробувань	Випробування завантаження файлів для підписування
Стан системи перед випробуванням	Система відкриває форму для працівника відділу (коодинатор, завідувач факультету/кафедри)
Вхідні дані	Назва файлу
Вихідні результати	Файли будуть зберігатися на локальному комп’ютері
Стан системи після випробувань	-

Висновок до розділу

В цьому розділу, розглядали основні форми для роботи системи, та було детально описано основні функції та інструкції для роботи з системою. Проведено випробування основних функціональностей системи, було розглянуто стан системи перед та після конкурсу, вхідні та вихідні результати.

ЗАГАЛЬНІ ВИСНОВКИ

В дипломному проекті детально розглядали проблеми процесу подачі документів в програмі європейських обмінів. Для участі в конкурсі, участники повинні подати документи до відділу мобільності та очікувати результати після декілька тижнів. Актуальність обгрунтовується тим, що система полегшує ці процеси і зменшує кількість часу на оформлення.

В першому розділі, було наведено опис предметного середовища, було обґрунтовано доцільність розробки системи підтримки навчання студентів - учасників програм європейських обмінів. Було сформульовано ціль та задачі розробки системи і було наведено існуючий аналог.Крім цього, було вказано принцип роботи окремих процесів і завдання для кожного актора та розроблені модель бізнес процесів та діаграма варіантів використання.

В другому розділі, ,було показані необхідні дані для подачі документів. Було наведено перелік сутностей в цьому проекті та детально розписано поля в кожній сутності, а також показано зв'язки між таблицями.

Було сформульовано постановка задачі математичного забезпечення, повна математична постановка задач. Мета задачі є обчислення рейтинг для окремих учасників як узагальненого фактора, який має найбільший вплив на результат конкурса. Для розвязку задачі використаний метод факторного аналіза

Було розроблено діаграму послідовності, яка показує які дії необхідно виконувати кожному актору та діаграма компонентів, яка показує основні компоненти в системі. Крім цього, було описано основні функції, їх призначення та параметри для функції.

В керівництві користувача детально описано основні функції та інструкції для процесу робот з системою. Також проведено випробування основних функціональностей системи, розглянуто стан системи перед та після конкурсу, вхідні та вихідні результати.

Завдання дипломного проекту виконано в повному обсязі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Офіційний сайт Еразмус в Україні «Про програму» [Електронний ресурс] // Режим доступу: https://erasmusplus.org.ua/erasmus/pro-prohramu.html
2. Офіційний сайт відділу академічної мобільності КПІ [Електронний ресурс] // Режим доступу: http://mobilnist.kpi.ua/competition/
3. Офіційний сайт відділу академічної мобільності КПІ [Електронний ресурс] // Режим доступу: http://mobilnist.kpi.ua/processing-documents/
4. Згуровський М.З., Коваленко І.І., Міхайленко В.М. Вступ до комп'ютерних інформаційних технологій:Навч. посібник.– К.: Вид-во Європ. ун-ту, 2003. –265 с.
5. Офіційний сайт Еразмус в Україні«Академічна мобільність» [Електронний ресурс] // Режим доступу: https://erasmusplus.org.ua/erasmus/ka1-navchalna-mobilnist.html
6. Боровиков В.П.Программа STATISTICA для студентов и инженеров. М.Компьютер Пресс, 2001-301с
7. Офіційний сайт Microsoft «Краткий обзор языка C#» [Електронний ресурс] // https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/
8. Офіційний сайт Microsoft «Требования к системе для семейства продуктов Visual Studio 2017» [Електронний ресурс] // Режим доступу: https://docs.microsoft.com/ru-ru/visualstudio/productinfo/vs2017-system-requirements-vs
9. Офіційний сайт METANIT«Введение в C#» [Електронний ресурс] // https://metanit.com/sharp/tutorial/1.1.php

Додаток А

Тексти програмного коду

Система підтримки навчання студентів – учасників європейських програм мобільності.

(Найменування програми (документа))

DVD-R

(Вид носія даних)

49 арк,8 Мб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020 року

Login.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace Erasmus
{
    public partial class Login : Form
    {
        public Login()
        {
            InitializeComponent();
        }
        string connectionString = "Data Source =.\\SQLEXPRESS;Initial Catalog = Public; Integrated Security = True;";
        private void Login_FormClosing(object sender, FormClosingEventArgs e)
        {
            if(MessageBox.Show("Do you want to exit?","Warning",MessageBoxButtons.OKCancel) !=
System.Windows.Forms.DialogResult.OK)
            {
                e.Cancel = true;
            }
        }
        private void entry_Click(object sender, EventArgs e)
        {
            string username = txtusername.Text;
            string password = txtpassword.Text;
            if (txtusername.Text == "")
            {
                MessageBox.Show("Немає логін", "Попередження", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                txtusername.Focus();
                return;
            }
            if (txtpassword.Text == "")
            {
                MessageBox.Show("Немає пароль", "Попередження", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                txtpassword.Focus();
                return;
            }
            try
            {
                CheckLogin(username, password);
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
             }
        public void CheckLogin(string username, string password)
                {
                    SqlConnection connection = new SqlConnection(connectionString);
                    SqlCommand sqlCommand = new SqlCommand("select Users.id, Users.username, Users.userlogin,
        Users.user_year, Users.user_role, Faculty.faculty_name from Users " +
                                                           "inner join Faculty on Users.faculty_id = Faculty.id " +
                                                           "WHERE userlogin = @login AND user_password = @password",
        connection);
                    SqlParameter uName = new SqlParameter("@login", SqlDbType.VarChar);
                    SqlParameter uPassword = new SqlParameter("@password", SqlDbType.VarChar);
                    uName.Value = username;
                    uPassword.Value = password;
                    sqlCommand.Parameters.Add(uName);
                    sqlCommand.Parameters.Add(uPassword);
                    sqlCommand.Connection.Open();
                    string username1;
                    string userfaculty;
                    string userlogin;
                    string user_year;
                    string user_role;
         int user_id;
            SqlDataReader myReader = sqlCommand.ExecuteReader(CommandBehavior.CloseConnection);
            if (myReader.Read() == true)
            {
                user_role = myReader["user_role"].ToString();
                user_id = Convert.ToInt32(myReader["id"]);
                username1 = myReader["username"].ToString();
                userfaculty = myReader["faculty_name"].ToString();
                if(myReader["user_year"] == null)
                {
                    user_year = "0";
                }
                else
```

```
                {

                    user_year = myReader["user_year"].ToString();
                }
                Exchange exchange = new Exchange(user_id, username1, userfaculty, user_year, user_role);
                exchange.Show();
            }
            else
            {
                MessageBox.Show("Аккаунт не існує", "Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Information);
                txtusername.Clear();
                txtpassword.Clear();
            }
            if (connection.State == ConnectionState.Open)
            {
                connection.Dispose();
            }
             }
        private void Login_Load(object sender, EventArgs e)
        {
            txtpassword.PasswordChar = '*';
        }

        private void createAccount_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            Registration registration = new Registration();
            registration.Show();
            this.Hide();
        }
    }
     }
```

Exchange.cs

```
using System;
using System.Data;
using System.IO;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace Erasmus
{
    public partial class Exchange : Form
    {
        static string connectionString = "Data Source =.\\SQLEXPRESS;Initial Catalog = Public; Integrated Security =
True;";
        SqlConnection connection = new SqlConnection(connectionString);
        string faculty;
        string username;
        string year;
        string role;
        int id;
        public Exchange(int id, string username, string faculty, string year, string role)
        {
            InitializeComponent();
            this.faculty = faculty;
            this.username = username;
            this.year = year;
            this.role = role;
            this.id = id;
        }
        private void аккаунтToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (role == "Студент")
            {
                User_Form user = new User_Form(id, username, faculty, year, role);
                user.Show();
            }
            else if(role == "Працівник відділу")
            {
                Employee employ = new Employee(id, username, role);
                employ.Show();
            }
            else if (role == "Координатор" || role == "Завідувач факультету/кафедри")
            {
                Professor pro = new Professor(id, username,faculty,role);
                pro.Show();
            }
        private void проАвторуToolStripMenuItem_Click(object sender, EventArgs e)
        {
            About about = new About();
            about.Show();
        }
```

```
        private void вихідToolStripMenuItem_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Do you want to exit?", "Warning", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);
            if (result == System.Windows.Forms.DialogResult.Yes){
                this.Close();
            }
        }
        private void Exchange_Load(object sender, EventArgs e)

        {

            string query = "select Exchange.id, Exchange.program, Exchange.country from Exchange";
            SqlCommand command = new SqlCommand(query, connection);
            connection.Open();
            DataTable dt = new DataTable();
            SqlDataAdapter da = new SqlDataAdapter(command);
            da.Fill(dt);
            list.DataSource = dt;
            int rowCount = (list.RowCount-1);
            for(int i = 0; i < rowCount; i++)
            {
                string item = list.Rows[i].Cells[1].Value.ToString();
                list_partner.Items.Add(item);
            }

            connection.Close();
        }
        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            Search_Univer();
        }
        private void документиToolStripMenuItem_Click(object sender, EventArgs e)

        {
            Document doc = new Document();
            doc.Show();
        }
           public void Search_Univer()
        {
            if (checkBox1.Checked)
            {
                string query = "select Exchange.id, Exchange.program, Exchange.country, Faculty.faculty_name,
Exchange.program_year" +
                               " from Exchange" +
                               " inner join Faculty" +
                               " on Exchange.faculty_id = Faculty.id " +
                               "where faculty_name = '" + faculty + "'";
                SqlCommand command = new SqlCommand(query, connection);
                connection.Open();
                DataTable dt = new DataTable();
                SqlDataAdapter da = new SqlDataAdapter(command);
                da.Fill(dt);
                list.DataSource = dt;
                connection.Close();
            } else
            {
                string query = "select Exchange.id, Exchange.program, Exchange.country, Faculty.faculty_name,
Exchange.program_year" +
                               " from Exchange" +
                               " inner join Faculty" +
                               " on Exchange.faculty_id = Faculty.id " +
                               "where faculty_name = '" + faculty + "'";
                SqlCommand command = new SqlCommand(query, connection);
                connection.Open();
                DataTable dt = new DataTable();
                SqlDataAdapter da = new SqlDataAdapter(command);
                da.Fill(dt);
                list.DataSource = dt;
                connection.Close();
            }
        }
    }
}
```

Registration.cs

```
using System;
using System.Data;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace Erasmus
{
```

```
    public partial class Registration : Form
    {
        public Registration()
        {
            InitializeComponent();
        }
        string connectionString = "Data Source=.\\sqlexpress;Initial Catalog=Public;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
        private void ok_Click(object sender, EventArgs e)
        {
            regis();
        }
        private void cancel_Click(object sender, EventArgs e)
        {
            Login sign_in = new Login();
            DialogResult result;
            result = MessageBox.Show("Ви впевнений?", "Попередження", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
            if (result == DialogResult.Yes)
            {
                this.Close();
                sign_in.Show();
            }
        }
        public void regis()
        {
            DialogResult result;
            int faculty_id = faculty.SelectedIndex + 1;
            int chair_id = chair.SelectedIndex + 1;
            int course = Convert.ToInt32(year.Value);
            if (name.Text == "" || login.Text == "" || faculty.Text == "" || chair.Text == "" || year.Text == "" ||
                role.Text == "" || password.Text == "" || rpassword.Text == "")
            {
                MessageBox.Show("Обов'язково заповнити всі поля", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else if (password.Text != rpassword.Text)
            {
                MessageBox.Show("Повторний пароль є неправильним", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            else
            {
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    string query = "INSERT INTO Users VALUES(@username, @userlogin, @faculty_id, @chair_id, @user_year,
@user_role, @user_password)";
                    using (SqlCommand queryInsertUser = new SqlCommand(query))
                    {
                           queryInsertUser.Connection = connection;
                           queryInsertUser.Parameters.Add("@username", SqlDbType.VarChar, 100).Value =
name.Text;
                           queryInsertUser.Parameters.Add("@userlogin", SqlDbType.VarChar, 100).Value =
login.Text;
                           queryInsertUser.Parameters.Add("@faculty_id", SqlDbType.Int).Value = faculty_id;
                           queryInsertUser.Parameters.Add("@chair_id", SqlDbType.Int).Value = chair_id;
                           queryInsertUser.Parameters.Add("@user_year", SqlDbType.Int).Value = course;
                           queryInsertUser.Parameters.Add("@user_role", SqlDbType.VarChar, 100).Value =
role.Text;
                           queryInsertUser.Parameters.Add("@user_password", SqlDbType.VarChar, 100).Value =
password.Text;
                           try
                           {
                               connection.Open();
                               queryInsertUser.ExecuteNonQuery();
                               result = MessageBox.Show("Успішно зареєстрував", "Повідомлення",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                               if (result == DialogResult.OK)
                               {
                                   name.Text = "";
                                   login.Text = "";
                                   faculty.Text = "";
                                   chair.Text = "";
                                   year.Text = "";
                                   role.Text = "";
                                    password.Text = "";
                                   rpassword.Text = "";
                               }
                           }
```

```
catch (SqlException ex)
                        {
                            MessageBox.Show(ex.Message, "Warning", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        }
                        finally
                        {
                            connection.Close();
                        }
                    }
                }
            }
        }
        private void Registration_Load(object sender, EventArgs e)
        {
            password.PasswordChar = '*';
            rpassword.PasswordChar = '*';
            if (Control.IsKeyLocked(Keys.CapsLock))
            {
                MessageBox.Show("The Caps Lock is ON", "Warning", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                password.Enabled = false;
                rpassword.Enabled = false;
            }
            else
            {
                password.Enabled = true;
                rpassword.Enabled = true;
            }
        }
    }
```

User_Form.cs

```
using System;
using System.Data;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.IO;
using System.Net;
namespace Erasmus
{
    public partial class User_Form : Form
    {
        string faculty_2;
        string username;
        string year;
        string role;
        int id;
        public string server = "ftp://192.168.56.1/Application";
        public string user_name = "PHAM HOANG";
        public string user_pass = "01101999hoang";
        private static string connectionString = "Data Source=.\\sqlexpress;Initial Catalog=Public;Integrated
Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
        public User_Form(int id, string username, string faculty, string year, string role)
        {
            InitializeComponent();
            this.username = username;
            this.faculty_2 = faculty;
            this.year = year;
            this.role = role;
            this.id = id;
        }
        private void проАвторуToolStripMenuItem_Click(object sender, EventArgs e)
        {
            About about = new About();
            about.Show();
        }
        private void User_Form_Load(object sender, EventArgs e)
        {
            username_1.Text = username;
            year_1.Text = year;
            role_1.Text = role;
            faculty_1.Text = faculty_2;
        }
        private void вихідToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }
```

```
        private void вихідToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }
         public void apply_document()
        {
            int publication_available = 0;
            int competition_available = 0;
            double average_score = Convert.ToDouble(average.Text);
            int exchangeid_1 = partner_1.SelectedIndex + 1;
            int exchangeid_2 = partner_2.SelectedIndex + 1;
            string fileName = Path.GetFileName(passport_scan.Text);
            string fileName6 = Path.GetFileName(avatar.Text);
            string fileName1 = Path.GetFileName(average_scan.Text);
            string fileName2 = Path.GetFileName(certificate_file.Text);
            string fileName3 = Path.GetFileName(publication_path.Text);
            string fileName4 = Path.GetFileName(competition_file.Text);
            string fileName5 = Path.GetFileName(learning_file.Text);
            if (publication_yes.Checked)
            {
                publication_available = 1;
                publication_path.Enabled = true;
                browse_publication.Enabled = true;
            }
            else
            {
                publication_available = 0;
                publication_path.Enabled = false;
                browse_publication.Enabled = false;

            }
            if (competition_yes.Checked)
            {
                competition_available = 1;
                competition_file.Enabled = true;
                browse_competition.Enabled = true;
            }
            else
            {
                competition_available = 0;
                competition_file.Enabled = false;
                browse_competition.Enabled = false;
            }
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
               string query = "INSERT INTO App_Docs VALUES(@id_student, @id_program, @passport_number, @passport_scan,
               @average_score, @average_scan, @english_skill, " +
                                  "@type_certificate, @certificate_scan, @publication_avail, @publication_file,
                                @competition_avail, @competition_file, " +
                               "@learning_agreement_file, @email, @avatar)";

                string query1 = "INSERT INTO App_Docs VALUES(@id_student, @id_program, @passport_number, @passport_scan,
@average_score, @average_scan, @english_skill, " +
                                "@type_certificate, @certificate_scan, @publication_avail, @publication_file,
@competition_avail, @competition_file, " +
                                "@learning_agreement_file, @email, @avatar)";
                using (SqlCommand queryInsert = new SqlCommand(query))
                {
                    queryInsert.Connection = connection;
                    queryInsert.Parameters.Add("@id_student", SqlDbType.Int).Value = id;
                    queryInsert.Parameters.Add("@id_program", SqlDbType.Int).Value = exchangeid_1;
                    queryInsert.Parameters.Add("@passport_number", SqlDbType.VarChar, 100).Value = series_pas.Text;
                    queryInsert.Parameters.Add("@passport_scan", SqlDbType.VarChar, 100).Value = fileName;
                    queryInsert.Parameters.Add("@average_score", SqlDbType.Float).Value = average_score;
queryInsert.Parameters.Add("@average_scan", SqlDbType.VarChar, 100).Value = fileName1;
                    queryInsert.Parameters.Add("@english_skill", SqlDbType.VarChar, 100).Value = language.Text;
                    queryInsert.Parameters.Add("@type_certificate", SqlDbType.VarChar, 100).Value = certificate.Text;
                    queryInsert.Parameters.Add("@certificate_scan", SqlDbType.VarChar, 100).Value = fileName2;
                    queryInsert.Parameters.Add("@publication_avail", SqlDbType.Int).Value = publication_available;
                    queryInsert.Parameters.Add("@publication_file", SqlDbType.VarChar, 100).Value = fileName3;
                    queryInsert.Parameters.Add("@competition_avail", SqlDbType.Int).Value = competition_available;
                    queryInsert.Parameters.Add("@competition_file", SqlDbType.VarChar, 100).Value = fileName4;
                    queryInsert.Parameters.Add("@learning_agreement_file", SqlDbType.VarChar, 100).Value = fileName5;
                    queryInsert.Parameters.Add("@email", SqlDbType.VarChar, 100).Value = email.Text;
                    queryInsert.Parameters.Add("@avatar", SqlDbType.VarChar, 100).Value = fileName6;
                    try
                    {
                        connection.Open();
                        queryInsert.ExecuteNonQuery();
                        connection.Close();

                    }
```

```
                    catch (SqlException ex)
                    {
                        MessageBox.Show(ex.Message, "Warning", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
                using (SqlCommand queryInsert1 = new SqlCommand(query1))
                {
                    queryInsert1.Connection = connection;
                    queryInsert1.Parameters.Add("@id_student", SqlDbType.Int).Value = id;
                    queryInsert1.Parameters.Add("@id_program", SqlDbType.Int).Value = exchangeid_2;
                    queryInsert1.Parameters.Add("@passport_number", SqlDbType.VarChar, 100).Value = series_pas.Text;
                    queryInsert1.Parameters.Add("@passport_scan", SqlDbType.VarChar, 100).Value = fileName;
                    queryInsert1.Parameters.Add("@average_score", SqlDbType.Float).Value = average_score;
                    queryInsert1.Parameters.Add("@average_scan", SqlDbType.VarChar, 100).Value = fileName1;
                    queryInsert1.Parameters.Add("@english_skill", SqlDbType.VarChar, 100).Value = language.Text;
                    queryInsert1.Parameters.Add("@type_certificate", SqlDbType.VarChar, 100).Value = certificate.Text;
                    queryInsert1.Parameters.Add("@certificate_scan", SqlDbType.VarChar, 100).Value = fileName2;
                    queryInsert1.Parameters.Add("@publication_avail", SqlDbType.Int).Value = publication_available;
queryInsert1.Parameters.Add("@competition_avail", SqlDbType.Int).Value = competition_available;
                    queryInsert1.Parameters.Add("@competition_file", SqlDbType.VarChar, 100).Value = fileName4;
                    queryInsert1.Parameters.Add("@learning_agreement_file", SqlDbType.VarChar, 100).Value = fileName5;
                    queryInsert1.Parameters.Add("@email", SqlDbType.VarChar, 100).Value = email.Text;
                    queryInsert1.Parameters.Add("@avatar", SqlDbType.VarChar, 100).Value = fileName6;
                    try
                    {
                        connection.Open();
                        queryInsert1.ExecuteNonQuery();
                        connection.Close();
                    }
                    catch (SqlException ex)
                    {
                        MessageBox.Show(ex.Message, "Warning", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }
        }
        public void UpLoadFile()
        {
            string FullName = this.passport_scan.Text;
            string FullName1 = this.average_scan.Text;
            string FullName2 = this.certificate_file.Text;
            string FullName3 = this.publication_path.Text;
            string FullName4 = this.competition_file.Text;
            string FullName5 = this.learning_file.Text;
                 string FullName6 = this.avatar.Text;

            FileInfo toUpload = new FileInfo(FullName);
            FileInfo toUpload1 = new FileInfo(FullName1);
            FileInfo toUpload2 = new FileInfo(FullName2);
            FileInfo toUpload3 = new FileInfo(FullName3);
            FileInfo toUpload4 = new FileInfo(FullName4);
            FileInfo toUpload5 = new FileInfo(FullName5);
            FileInfo toUpload6 = new FileInfo(FullName6);
            //================Upload passport scan=================//
            FtpWebRequest request = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload.Name);
            request.Method = WebRequestMethods.Ftp.UploadFile;
            request.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
            Stream stream = request.GetRequestStream();
            FileStream fs = File.OpenRead(FullName);
            byte[] buffer = new byte[1024];
            double total = (double)fs.Length;
    int byteRead = 0;

            do
            {
                byteRead = fs.Read(buffer, 0, 1024);
                stream.Write(buffer, 0, byteRead);

            } while (byteRead != 0);
            fs.Close();
            stream.Close();
            //================Upload Average Score=================//
            FtpWebRequest request1 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload1.Name);
            request1.Method = WebRequestMethods.Ftp.UploadFile;
            request1.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
            Stream stream1 = request1.GetRequestStream();
            FileStream fs1 = File.OpenRead(FullName1);
            byte[] buffer1 = new byte[1024];
            double total1 = (double)fs1.Length;
```

```
        int byteRead1 = 0;
                //double read = 0;
                do
                {
                    byteRead1 = fs1.Read(buffer1, 0, 1024);
                    stream1.Write(buffer1, 0, byteRead1);
                    //read += (double)byteRead;

                } while (byteRead1 != 0);
                fs1.Close();
                stream1.Close();
                //================Upload Certificate=================//
                FtpWebRequest request2 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload2.Name);
                request2.Method = WebRequestMethods.Ftp.UploadFile;
                request2.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
                Stream stream2 = request2.GetRequestStream();
                FileStream fs2 = File.OpenRead(FullName2);
                byte[] buffer2 = new byte[1024];
                double total2 = (double)fs2.Length;
                int byteRead2 = 0;
                do
                {
                    byteRead2 = fs2.Read(buffer2, 0, 1024);
                    stream2.Write(buffer2, 0, byteRead2);
                    //read += (double)byteRead;

                     } while (byteRead2 != 0);

                fs2.Close();
                stream2.Close();
                //================Upload Publication=================//
                FtpWebRequest request3 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload3.Name);
                request3.Method = WebRequestMethods.Ftp.UploadFile;
                request3.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
                Stream stream3 = request3.GetRequestStream();
            FileStream fs3 = File.OpenRead(FullName3);

                byte[] buffer3 = new byte[1024];
                double total3 = (double)fs3.Length;
                int byteRead3 = 0
        do
                {
                    byteRead3 = fs3.Read(buffer3, 0, 1024);
                    stream3.Write(buffer3, 0, byteRead3);
                    //read += (double)byteRead;

                } while (byteRead3 != 0);
                fs3.Close();
                stream3.Close();
                //================Upload Competition=================//
                FtpWebRequest request4 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload4.Name);
                request4.Method = WebRequestMethods.Ftp.UploadFile;
                request4.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
                Stream stream4 = request4.GetRequestStream();
                FileStream fs4 = File.OpenRead(FullName4);
                byte[] buffer4 = new byte[1024];
                double total4 = (double)fs4.Length;
                int byteRead4 = 0;
                do
                {
                    byteRead4 = fs4.Read(buffer4, 0, 1024);
                    stream4.Write(buffer4, 0, byteRead4);
                    //read += (double)byteRead;

                } while (byteRead4 != 0);
                fs4.Close();
                stream4.Close();
                //================Upload Learning Agreement=================//
                FtpWebRequest request5 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload5.Name);
                request5.Method = WebRequestMethods.Ftp.UploadFile;
                request5.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
                Stream stream5 = request5.GetRequestStream();
                FileStream fs5 = File.OpenRead(FullName5);
                byte[] buffer5 = new byte[1024];
                double total5 = (double)fs5.Length;
                int byteRead5 = 0;
        do
                {
                    byteRead5 = fs5.Read(buffer5, 0, 1024);
                    stream5.Write(buffer5, 0, byteRead5);
                    //read += (double)byteRead;
                } while (byteRead5 != 0);
                fs5.Close();
            stream5.Close();
```

```
            FtpWebRequest request6 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload6.Name);
            request6.Method = WebRequestMethods.Ftp.UploadFile;
            request6.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
            Stream stream6 = request6.GetRequestStream();
            FileStream fs6 = File.OpenRead(FullName6);
            byte[] buffer6 = new byte[1024];
            double total6 = (double)fs6.Length;
            int byteRead6 = 0;
            do
            {
                byteRead6 = fs6.Read(buffer6, 0, 1024);
                     stream6.Write(buffer6, 0, byteRead6);

            } while (byteRead6 != 0);
            fs6.Close();
            stream6.Close();
        }
        private void apply_Click(object sender, EventArgs e)
        {
            UpLoadFile();
            apply_document();
            DialogResult result = MessageBox.Show("Успішно зареєстрував", "Повідомлення", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            if(result == DialogResult.OK)
            {
                avatar.Text = "";
                series_pas.Text = "";
                passport_scan.Text = "";
                email.Text = "";
                average.Text = "";
                average_scan.Text = "";
                partner_1.Text = "";
                partner_2.Text = "";
                language.Text = "";
                certificate_file.Text = "";
                certificate.Text = "";
                publication_path.Text = "";
                competition_file.Text = "";
                learning_file.Text = "";
            }
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Form1 form1 = new Form1();
            form1.Show();
        }
         private void browse_certificate_Click(object sender, EventArgs e)
        {
            //FileInfo file;
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg; *.jpe; *.jfif; *.png" })
            {
                if(openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    certificate_file.Text = filename;
}
            }
        }
private void browse_publication_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Document files (*.doc, *.docx, *.txt, *pdf) | *.doc; *.docx; *.txt; *.pdf" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    publication_path.Text = filename
}
            }
        }
        private void browse_competition_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Document files (*.doc, *.docx, *.txt, *pdf) | *.doc; *.docx; *.txt; *.pdf" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    competition_file.Text = filename;
                }
            }
        }
```

```
        private void learning_browse_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "All files|*.*" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    learning_file.Text = filename;
                }
            }
        }
        private void average_browse_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg; *.jpe; *.jfif; *.png" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    average_scan.Text = filename;
                }
            }
        }
        private void browse_passport_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg; *.jpe; *.jfif; *.png"})
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    passport_scan.Text = filename;
                }
            }
        }
        private void avatar_browse_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg; *.jpe; *.jfif; *.png" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    avatar.Text = filename;
                }
            }
        }
        private void zayava_browse_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Document files (*.doc, *.docx, *.txt, *pdf) | *.doc; *.docx; *.txt; *.pdf" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    competition_file.Text = filename;
                }
            }
        }
        private void invitation_browse_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Document files (*.doc, *.docx, *.txt, *pdf) | *.doc; *.docx; *.txt; *.pdf" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    competition_file.Text = filename;
                }
            }
        }

        private void invitation_trans_browse_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Document files (*.doc, *.docx, *.txt, *pdf) | *.doc; *.docx; *.txt; *.pdf" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    competition_file.Text = filename;

                }
            }
        }
```

```
        private void contract_browse_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Document files (*.doc, *.docx, *.txt, *pdf) | *.doc; *.docx; *.txt; *.pdf" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    competition_file.Text = filename;
                }
            }
        }
        private void apply_1_Click(object sender, EventArgs e)
        {
            UploadFile_1();
            Apply_1_etap();
            DialogResult result = MessageBox.Show("Успішно подав документи", "Повідомлення", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            if (result == DialogResult.OK)
            {
                zayava.Text = "";
                invitation.Text = "";
                invitation_trans.Text = "";
                contract.Text = "";
            }
        }
        public void Apply_1_etap()
        {
            string fileName1 = Path.GetFileName(zayava.Text);
            string fileName2 = Path.GetFileName(invitation.Text);
            string fileName3 = Path.GetFileName(invitation_trans.Text);
            string fileName4 = Path.GetFileName(contract.Text);

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string query = "INSERT INTO Doument VALUES(@id_student, @application, @invitation, @invitation_trans,
@contract)";
                using (SqlCommand queryInsert = new SqlCommand(query))
                {
                    queryInsert.Connection = connection;
                    queryInsert.Parameters.Add("@id_student", SqlDbType.Int).Value = id;

                    queryInsert.Parameters.Add("@application", SqlDbType.VarChar, 100).Value = fileName1;
                    queryInsert.Parameters.Add("@invitation", SqlDbType.VarChar, 100).Value = fileName2;

                    queryInsert.Parameters.Add("@invitation_trans", SqlDbType.VarChar, 100).Value = fileName3;
                    queryInsert.Parameters.Add("@contract", SqlDbType.VarChar, 100).Value = fileName4;
                }
            }
        }
        public void UploadFile_1()
        {
            string FullName = this.zayava.Text;
            string FullName1 = this.invitation.Text;
            string FullName2 = this.invitation_trans.Text;
            string FullName3 = this.contract.Text;
            FileInfo toUpload = new FileInfo(FullName);
            FileInfo toUpload1 = new FileInfo(FullName1);
            FileInfo toUpload2 = new FileInfo(FullName2);
            FileInfo toUpload3 = new FileInfo(FullName3);
            //================Upload passport scan=================//
            FtpWebRequest request = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload.Name);
            request.Method = WebRequestMethods.Ftp.UploadFile;
            request.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
            Stream stream = request.GetRequestStream();
            FileStream fs = File.OpenRead(FullName);
            byte[] buffer = new byte[1024];
            double total = (double)fs.Length;
            int byteRead = 0;
            //double read = 0;
            do
            {
                byteRead = fs.Read(buffer, 0, 1024);
                stream.Write(buffer, 0, byteRead);
} while (byteRead != 0);
            fs.Close();
            stream.Close();
```

```
//================Upload Average Score=================//
            FtpWebRequest request1 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload1.Name);
            request1.Method = WebRequestMethods.Ftp.UploadFile;
            request1.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
            Stream stream1 = request1.GetRequestStream();
            FileStream fs1 = File.OpenRead(FullName1);
            byte[] buffer1 = new byte[1024];
            double total1 = (double)fs1.Length;
            int byteRead1 = 0;
            //double read = 0;
            do
            {
                byteRead1 = fs1.Read(buffer1, 0, 1024);
                stream1.Write(buffer1, 0, byteRead1);
            } while (byteRead1 != 0);
            fs1.Close();
            stream1.Close();
            FtpWebRequest request2 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload2.Name);
            request2.Method = WebRequestMethods.Ftp.UploadFile;
            request2.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
            Stream stream2 = request2.GetRequestStream();
            FileStream fs2 = File.OpenRead(FullName2);
            byte[] buffer2 = new byte[1024];
            double total2 = (double)fs2.Length;
            int byteRead2 = 0;
            do
            {
                byteRead2 = fs2.Read(buffer2, 0, 1024);
                stream2.Write(buffer2, 0, byteRead2);
            } while (byteRead2 != 0);
fs2.Close();
            stream2.Close();
            //================Upload Publication=================//
            FtpWebRequest request3 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload3.Name);
            request3.Method = WebRequestMethods.Ftp.UploadFile;
            request3.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
            Stream stream3 = request3.GetRequestStream();
            FileStream fs3 = File.OpenRead(FullName3);
            byte[] buffer3 = new byte[1024];
            double total3 = (double)fs3.Length;
            int byteRead3 = 0;
            do
            {
                byteRead3 = fs3.Read(buffer3, 0, 1024);
                stream3.Write(buffer3, 0, byteRead3);
} while (byteRead3 != 0);
            fs3.Close();
            stream3.Close();
        }
        private void ipn_browse_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Document files (*.doc, *.docx, *.txt, *pdf) | *.doc; *.docx; *.txt; *.pdf" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    competition_file.Text = filename;
                }
            }
        }
        private void nakaz_browse_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Document files (*.doc, *.docx, *.txt, *pdf) | *.doc; *.docx; *.txt; *.pdf" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
                    competition_file.Text = filename;
                }
            }
        }
         public void UploadFile_2()
        {
            string FullName = this.ipn.Text;
            string FullName1 = this.nakaz.Text;

            FileInfo toUpload = new FileInfo(FullName);
            FileInfo toUpload1 = new FileInfo(FullName1);
```

```
            FtpWebRequest request = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload.Name);
            request.Method = WebRequestMethods.Ftp.UploadFile;
            request.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
            Stream stream = request.GetRequestStream();
            FileStream fs = File.OpenRead(FullName);
            byte[] buffer = new byte[1024];
            double total = (double)fs.Length;
            int byteRead = 0;
            //double read = 0;
            do
            {
                byteRead = fs.Read(buffer, 0, 1024);
                stream.Write(buffer, 0, byteRead);
            } while (byteRead != 0);
            fs.Close();
            stream.Close();
            //================Upload Average Score=================//
            FtpWebRequest request1 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload1.Name);
            request1.Method = WebRequestMethods.Ftp.UploadFile;
            request1.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
            Stream stream1 = request1.GetRequestStream();
            FileStream fs1 = File.OpenRead(FullName1);
            byte[] buffer1 = new byte[1024];
            double total1 = (double)fs1.Length;
            int byteRead1 = 0;
            //double read = 0;
            do
            {
                byteRead1 = fs1.Read(buffer1, 0, 1024);
                stream1.Write(buffer1, 0, byteRead1);
                //read += (double)byteRead;

            } while (byteRead1 != 0);
            fs1.Close();
            stream1.Close();
        }
        public void Apply_2_etap()
        {
            string fileName1 = Path.GetFileName(ipn.Text);
            string fileName2 = Path.GetFileName(nakaz.Text);
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string query = "INSERT INTO Doument VALUES(@inp, @nakaz)";
                using (SqlCommand queryInsert = new SqlCommand(query))
                {
                    queryInsert.Connection = connection;
                    queryInsert.Parameters.Add("@inp", SqlDbType.VarChar, 100).Value = fileName1;
                    queryInsert.Parameters.Add("@nakaz", SqlDbType.VarChar, 100).Value = fileName2;
                }
            }
        }

       private void button3_Click(object sender, EventArgs e)
        {
            UploadFile_2();
            Apply_2_etap();
            DialogResult result = MessageBox.Show("Успішно подав документи", "Повідомлення", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            if (result == DialogResult.OK)
            {
                ipn.Text = "";
                nakaz.Text = "";

            }
        }
    }
     }
```

Employee.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Net;
using System.IO;
namespace Erasmus
{
    public partial class Employee : Form
    {
        //string faculty_2;
        string username;
        string role;
```

```
public string server = "ftp://192.168.56.1/Application";
        SqlConnection connection = new SqlConnection(connectionString);
        public string user_name = "PHAM HOANG";
        public string user_pass = "01101999hoang";
        static string connectionString = "Data Source =.\\SQLEXPRESS;Initial Catalog = Public; Integrated Security =
True;";
        public Employee(int id, string username, string role)
        {
            InitializeComponent();
            this.username = username;
            this.role = role;
            this.id = id;
        }
        private void проАвтораToolStripMenuItem_Click(object sender, EventArgs e)
        {
            About about = new About();
            about.Show();
          }

    private void button1_Click(object sender, EventArgs e)
        {
            string partner = "";
            partner = partner_1.Text;
            string query = "select Users.username, Exchange.program, App_Docs.email from App_Docs" +
                           " inner join Users" +
                           " on App_Docs.id_student = Users.id" +
                           " inner join Exchange" +
                           " on App_Docs.id_program = Exchange.id where Exchange.program = '"+partner+"'";
            SqlCommand command = new SqlCommand(query, connection);
            connection.Open();
            DataTable dt = new DataTable();
            SqlDataAdapter da = new SqlDataAdapter(command);
            da.Fill(dt);
            list.DataSource = dt;
            connection.Close();
        }
         private void button2_Click(object sender, EventArgs e)
        {
            Form1 form1 = new Form1();
            form1.Show();
        }
private void Employee_Load(object sender, EventArgs e)
        {
            //SqlConnection connection = new SqlConnection(connectionString);
            username_1.Text = username;
            role_1.Text = role;
            string query = "select Users.username, Exchange.program, a.passport_number, a.passport_scan, a.average_score,
a.average_score_scan, " +
                "a.english_skill, a.type_certificate, a.certificate_scan, " +
                "publication_file, a.competition_file, a.learning_agreement_file, a.avatar " +
                "from App_Docs as a inner join Users " +
                "on a.id_student = Users.id inner join Exchange " +
                "on a.id_program = Exchange.id";
            SqlCommand command = new SqlCommand(query, connection);
            connection.Open();
            DataTable dt = new DataTable();
            SqlDataAdapter da = new SqlDataAdapter(command);
            da.Fill(dt);
            list.DataSource = dt;
            connection.Close();
            SqlConnection connection1 = new SqlConnection(connectionString);
            string query1 = "select App_Docs.id_student, Users.username, Exchange.program, App_Docs.average_score,
App_Docs.english_skill, App_Docs.type_certificate," +
                " App_Docs.publication_score, App_Docs.competition_score, App_Docs.rating" +
                " from App_Docs" +
                " inner join Users" +
                " on App_Docs.id_student = Users.id" +
                " inner join Exchange" +
                " on App_Docs.id_program = Exchange.id";
            SqlCommand command1 = new SqlCommand(query1, connection1);
            connection1.Open();
            DataTable dt1 = new DataTable();
            SqlDataAdapter da1 = new SqlDataAdapter(command1);
            da1.Fill(dt1);
            listing.DataSource = dt1;
            connection1.Close();
        }
private void button1_Click(object sender, EventArgs e)
        {
            string partner = "";
            partner = partner_1.Text;
            string query = "select Users.username, Exchange.program, App_Docs.email from App_Docs" +
                           " inner join Users" +
                           " on App_Docs.id_student = Users.id" +
```

```
" inner join Exchange" +
                            " on App_Docs.id_program = Exchange.id where Exchange.program = '"+partner+"'";
            SqlCommand command = new SqlCommand(query, connection);
            connection.Open();
            DataTable dt = new DataTable();
            SqlDataAdapter da = new SqlDataAdapter(command);
            da.Fill(dt);
            list.DataSource = dt;
            connection.Close();
             }

        private void download_Click(object sender, EventArgs e)
        {
            List<int> list_ind = new List<int>() {3, 5, 8, 9, 10, 11, 12};
            string path = @"C:\Users\famsu\Desktop\";
            string folder;

            for (int i = 1; i < list.RowCount; i = i + 2)
            {
                folder = list.Rows[i].Cells[0].Value.ToString();
                System.IO.Directory.CreateDirectory(path + folder);
                foreach (int j in list_ind)
                {
                    string fileName = list.Rows[i].Cells[j].Value.ToString();
                    Download(folder, fileName);
                                }
            }
            MessageBox.Show("Successful download!", "Information", MessageBoxButtons.OK);
        }
        public void Download(string folder, string fileName)
        {
            try
            {
                FtpWebRequest request = (FtpWebRequest)WebRequest.Create(server + "/" + fileName);
                request.Method = WebRequestMethods.Ftp.DownloadFile;
                request.Credentials = new NetworkCredential(user_name, user_pass);
                FtpWebResponse response = (FtpWebResponse)request.GetResponse();
                string folder_1 = folder + @"\";
                string path = @"C:\Users\famsu\Desktop\" + folder_1;
                using (Stream responseStream = response.GetResponseStream())
                {
                    using (Stream fileStream = new FileStream(path + fileName, FileMode.CreateNew))
                    {
                        responseStream.CopyTo(fileStream);
                    }
                }
            }catch(WebException ex)
            {
                throw new Exception((ex.Response as FtpWebResponse).StatusDescription);
            }
          public void Scoring()
        {
            string participant = this.participant.Text;
            int publication_score = Convert.ToInt32(pub_score.Text);
            int competition_score = Convert.ToInt32(comp_score.Text);
            int coef = 0;
            int english = 0;
            for (int i = 0; i < listing.RowCount-1; i++)
            {
                if(listing.Rows[i].Cells[4].Value.ToString() == "B1")
                {
                    english = 1;
                }
                else if(listing.Rows[i].Cells[4].Value.ToString() == "B2")
                {
                    english = 2;
                }
                else if(listing.Rows[i].Cell{
                    english = 3;
                }
                else
                {
                    english = 4;
                }
            }
```

```
            for (int i = 0; i < listing.RowCount - 1; i++)
            {
                if (listing.Rows[i].Cells[5].Value.ToString() == "University")
                {
                    coef = 1;
                }
                else
                {
                    coef = 2;
                }
            }
            int english_score = english * coef;
            double rating;
            uint index;
            for (int i = 0; i < listing.RowCount - 1; i++)
            {
                if(participant == listing.Rows[i].Cells[1].Value.ToString())
                {
                    listing.Rows[i].Cells[6].Value = publication_score;
                    listing.Rows[i].Cells[7].Value = competition_score;
                    rating = Convert.ToDouble(listing.Rows[i].Cells[3].Value.ToString()) * 0.1 + english_score +
Convert.ToUInt32(listing.Rows[i].Cells[6].Value.ToString()) +
Convert.ToUInt32(listing.Rows[i].Cells[7].Value.ToString());
                    listing.Rows[i].Cells[8].Value = rating;
                    index = Convert.ToUInt32(listing.Rows[i].Cells[0].Value.ToString());
                    string query = @"update App_Docs set publication_score ='" + publication_score + "',
competition_score = '" + competition_score + "', rating = '"+rating+"' where id_student = '" +index+ "'";
                    using(SqlConnection connection = new SqlConnection(connectionString))
                    {
                        using (SqlCommand command = new SqlCommand())
                        {
                            command.Connection = connection;
                            connection.Open();
                            command.CommandText = query;
                            command.ExecuteNonQuery();
                        }
                    }
                }
            }
            MessageBox.Show("Successful edit", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);
             }
```

Server.cs

```
using System;
using System.Collections;
using LanMessengerLib;
namespace Temporary_Server
{
    public class Server : MarshalByRefObject, LanMessengerLib.IServer
    {
        Database database = new Database();
        [Serializable]
        public class Letter
        {
            string from;
            string to;
            string message;

            public Letter(string from, string to, string message)
            {
                this.from = from;
                this.to = to;
                this.message = message;
            }

            public string From
            {
                get
                {
                    return from;
                }
            }
          public string To
            {
                get
                {
                    return to;
                }
            }
```

```
public string Message
            {
                get
                {
                    return message;
                }
            }
             }

     ArrayList letters = new ArrayList();
        public Server()
        {
            database.Load();
        }
        public bool SignUp(string username, string password, string fullName, string IP)
        {
            return database.Add(username, password, fullName, IP);
        }
        public bool ChangePassword(string username, string curPassword, string newPassword)
        {
            return database.ChangePassword(username, curPassword, newPassword);
     }

         public bool SignIn(string username, string password, bool visible)
         {
             return database.SignIn(username, password, visible);
         }
         public bool SignOut(string username)
         {
             return database.SignOut(username);
         }
         public bool IsLoggedIn(string username)
         {
             return database.IsLoggedIn(username);
         }
         public bool IsVisible(string username)
         {
             return database.IsVisible(username);
         }
         public void ChangeStatus(string username)
         {
             database.ChangeStatus(username);
         }
         public void ChangeDisplayName(string username, string displayName)
         {
             database.ChangeDisplayName(username, displayName);
         }
         public string GetfullName(string username)
         {
             return database.GetfullName(username);
         }
         public void SetIP(string username, string IP)
         {
             database.SetIP(username, IP);
          }
          public string GetIP(string username)
         {
             return database.GetIP(username);
         }
         public bool AddContact(string username, string contact)
         {
             return database.AddContact(username, contact);
         }
         public bool RemoveContact(string username, string contact)
         {
             return database.RemoveContact(username, contact);
              }

          public ArrayList GetContacts(string username)
         {
             return database.GetContacts(username);
         }

              private const int maxOfflineMessages = 100;
```

```
public bool Send(string from, string to, string message)
        {
            if (database.ContactExists(to) == false)
                return false;
            if (database.IsLoggedIn(to))
                letters.Add(new Letter(from, to, message));
            else
            {
                database.offlineMessages.Add(new Letter(from, to, message));
                if (database.offlineMessages.Count > maxOfflineMessages)
                {
                    database.offlineMessages.RemoveAt(0);
                }
                Console.WriteLine(DateTime.Now.ToString() + ": sent Offline message to " + to);
            }
            return true;
        }
        public ArrayList ReceiveOffline(string to)
        {
            int length = database.offlineMessages.Count;
            ArrayList offlines = new ArrayList();
            for (int i = 0; i < length; i++)
            {
                if ((database.offlineMessages[i] as Letter).To == to)
                {
                    offlines.Add(new LetterReceive((database.offlineMessages[i] as Letter).Message,
(database.offlineMessages[i] as Letter).From));
                    database.offlineMessages.RemoveAt(i);
                    i--;
                    length--;
                    continue;
                }
            }
            return offlines;
        }
        public LetterReceive Receive(string to)
        {
            int length = letters.Count;
            for (int i = 0; i < length; i++)
            {
                if (((Letter)letters[i]).To == to)
                {
                    LetterReceive lr = new LetterReceive(((Letter)letters[i]).Message, ((Letter)letters[i]).From);
                    letters.RemoveAt(i);
                    return lr;
                }
            }
            return new LetterReceive("", "");
        }
    }
}
```

Database.cs

```
using System;
using System.Collections;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
namespace Temporary_Server
{
    [Serializable]
    class AccountField
    {
        internal string status;
        internal string password;
        internal string fullName;
        internal string IP;
        internal bool isLoggedIn;
        internal bool visible;
        internal ArrayList contacts;

        public AccountField(string password, bool isLoggedIn, string fullName, string IP)
        {
            this.fullName = fullName;
            this.password = password;
            this.isLoggedIn = isLoggedIn;
            this.IP = IP;
            contacts = new ArrayList();
        }
    }
    [Serializable]
class Database
    {
        private Hashtable database;
        internal ArrayList offlineMessages;
```

```
        public Database()
        {
            database = new Hashtable();
            offlineMessages = new ArrayList();
            Load();
        }
        internal bool Add(string username, string password, string fullName, string IP)
        {
            if (database.Contains(username))
                return false;
            database.Add(username, new AccountField(password, false, fullName, IP));
            return Save();
        }
     internal bool ChangePassword(string username, string curPassword, string newPassword)
        {
            if (database.Contains(username))
            {
                if (((AccountField)database[username]).password == curPassword)
                {
                    ((AccountField)database[username]).password = newPassword;
                    Save();
                    return true;
                }
                else
                {
                    return false
      else
                 {
                     return false;
                 }
      }
internal bool ChangeDisplayName(string username, string displayName)
        {
            if (database.Contains(username))
            {
                ((AccountField)database[username]).fullName = displayName;
                return true;
            }
            else
            {
                return false;
            }
        }
        internal bool SetIP(string username, string IP)
        {
            if (database.Contains(username))
            {
                ((AccountField)database[username]).IP = IP;
                return true;
            }
            else
            {
                return false;
            }
        }
internal bool SignIn(string username, string password, bool visible)
        {
            if (database.Contains(username))
            {
                if (((AccountField)database[username]).password == password)
                {
                    ((AccountField)database[username]).isLoggedIn = true;
                    ((AccountField)database[username]).visible = visible;
                    Console.WriteLine(DateTime.Now.ToString() + ": " + username + " successful sign in!");
                    return true;
                }
                else
                {
                    return false;
                }
            }
            return false;
             }

internal bool SignOut(string username)
        {
            if (database.Contains(username))
            {
                ((AccountField)database[username]).isLoggedIn = false;
                ((AccountField)database[username]).visible = false;
                Console.WriteLine(DateTime.Now.ToString() + ": " + username + " signed out!");
                return true;
            }
            return false;
             }
```

```
internal bool ContactExists(string username)
        {
            return (database.Contains(username));
        }
        internal bool IsLoggedIn(string username)
        {
            if (database.Contains(username))
            {
                return ((AccountField)database[username]).isLoggedIn;
            }
            else
            {
                return false;
            }
        }
        internal bool IsVisible(string username)
        {
            if (database.Contains(username))
            {
                return ((AccountField)database[username]).visible;
            }
            else
            {
                return false;
            }
        }
        internal void ChangeStatus(string username)
        {
            ((AccountField)database[username]).visible = !((AccountField)database[username]).visible;
        }
        internal string GetfullName(string username)
        {
            return ((AccountField)database[username]).fullName;
        }
        internal string GetIP(string username)
        {
            return ((AccountField)database[username]).IP;
             }

internal bool AddContact(string username, string contact)
        {
            if (database.Contains(username))
            {
                if (((AccountField)database[username]).contacts.Contains(contact))
                    return false;
                if (database.Contains(contact))
                {
                    ((AccountField)database[username]).contacts.Add(contact);
                    return Save();
                }
            }
            return false;
             }

internal bool RemoveContact(string username, string contact)
        {
            if (database.Contains(username))
            {
                if (((AccountField)database[username]).contacts.Contains(contact)) {
                    ((AccountField)database[username]).contacts.Remove(contact);
                    return Save();
                }
            }
            return false;
        }
        internal ArrayList GetContacts(string username)
        {
            if (database.Contains(username))
            {
                return ((AccountField)database[username]).contacts;
            }
            return null;
        }
        internal bool Load()
        {
            try
            {
                FileStream fileStream = new FileStream("UserInfo.dat", FileMode.Open);
                BinaryFormatter binaryFormatter = new BinaryFormatter();
                Database temp = (Database)binaryFormatter.Deserialize(fileStream);
                this.database = temp.database;
```

```
            this.offlineMessages = temp.offlineMessages;
                fileStream.Close();
                return true;
            }
            catch
            {
                database = new Hashtable();
                offlineMessages = new ArrayList();
                return false;
            }
        }
        internal bool Save()
        {
            try
            {
                FileStream fileStream =
                    new FileStream("UserInfo.dat", FileMode.Create);
                BinaryFormatter binaryFormatter = new BinaryFormatter();
                binaryFormatter.Serialize(fileStream, this);
                fileStream.Close();
                return true;
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
                return false;
            }
        }
    }
     }
```

LetterReceive.cs

```
using System;
using System.Collections;
namespace LanMessengerLib
{
    [Serializable]
    public class LetterReceive
    {
        string message;
        string from;

        public LetterReceive(string message, string from)
        {
            this.message = message;
            this.from = from;
        }

        public string Message
        {
            get
            {
                return message;
            }
        }

        public string From
        {
            get
            {
                return from;
            }
        }
    }
public interface IServer
    {
        bool SignUp(string username, string password, string fullName, string IP);
        bool ChangePassword(string username, string curPassword, string newPassword);
        bool SignIn(string username, string password, bool visible);
        bool SignOut(string username);
        bool IsVisible(string username);
        bool AddContact(string username, string contact);
        bool RemoveContact(string username, string contact);
        void ChangeStatus(string username);
        void ChangeDisplayName(string username, string displayName);
string GetfullName(string username);
        string GetIP(string username);
        void SetIP(string username, string IP);
        ArrayList GetContacts(string username);
bool Send(string from, string to, string message);
        LetterReceive Receive(string to);
        ArrayList ReceiveOffline(string to);
    }
     }
```

TrayBalloon.cs

```
sing System;
using System.Runtime.InteropServices;
namespace TrayBalloon
{
    public class TrayBalloon : IDisposable
    {
        private readonly TrayBalloonFrm Frm;

        private delegate void RunDialogHandler();

        private readonly static System.Collections.Queue CurrentlyVisible;
        static TrayBalloon()
        {
            CurrentlyVisible =
                System.Collections.Queue.Synchronized(new System.Collections.Queue());
        }

        public TrayBalloon()
        {
            Frm = new TrayBalloonFrm();
        }

        public string Message
        {
            get
            {
                return Frm.Message;
            }

            set
            {
                Frm.Message = value;
            }
             }


         public bool UseOpacity
        {
            get
            {
                return Frm.UseOpacity;
            }
            set
            {
                Frm.UseOpacity = value;
            }
        }

        public bool TopMost
        {
            get
            {
                return Frm.TopMost;
            }
            set
            {
                Frm.TopMost = value;
            }
             }

        public bool LightWeight
        {
            get
            {
                return Frm.LightWeight;
            }
            set
            {
                Frm.LightWeight = value;
            }
        }
        public string Title
        {
            get
            {
                return Frm.Title;
            }

            set
            {
                Frm.Title = value;
            }
        }
```

```
public string SoundLocation
        {
            get
            {
                return Frm.SoundLocation;
            }
            set
            {
                Frm.SoundLocation = value;
            }
        }
        public string BackgroundLocation
        {
            get
            {
                return Frm.BackgroundLocation;
            }
            set
            {
                Frm.BackgroundLocation = value;
            }
        }
        private static void Unregister()
        {
            lock (CurrentlyVisible)
            {
                CurrentlyVisible.Dequeue();
            }
             }

         private int GetFreeCount()
        {
            int Full = System.Windows.Forms.SystemInformation.WorkingArea.Height;
            return Full / Frm.Height;
             }

        private int RegisterAndStartingOffsetIndex()
        {
            lock (CurrentlyVisible)
            {
                CurrentlyVisible.Enqueue(Frm);

                if (CurrentlyVisible.Count <= 1)
                    return 0;

                bool[] Poss = new bool[GetFreeCount()];
                for (int Idx = 0; Idx < Poss.Length; Idx++)
                    Poss[Idx] = true;

                foreach (TrayBalloonFrm XFrm in CurrentlyVisible.ToArray())
                {
                    if (!(XFrm == Frm))
                        if (XFrm.StartingOffsetIndex < Poss.Length)
                            Poss[XFrm.StartingOffsetIndex] = false;
                }
                for (int Idx = 0; Idx < Poss.Length; Idx++)
                    if (Poss[Idx] == true)
                        return Idx;
                return Poss.Length - 1;
            }
        }
        [DllImport("user32.dll", CharSet = CharSet.Auto, ExactSpelling = true)]
        public static extern bool SetWindowPos(IntPtr hWnd, IntPtr hWndInsertAfter, int x, int y, int cx, int cy, int
flags);

        private void RunDialog()
        {
            Frm.StartingOffsetIndex = RegisterAndStartingOffsetIndex();

            SetWindowPos(Frm.Handle, (IntPtr)(-1), 0, 0, 0, 0, 0x50);

            Frm.ShowDialog();
            Unregister();
        }
        public void Run()
        {
            RunDialogHandler RDH = new RunDialogHandler(RunDialog);
            RDH.BeginInvoke(null, null);
             }

         public void Run(string Message)
        {
            this.Message = Message;
            Run();
        }
```

```
        public void Run(string Title, string Message)
        {
            this.Title = Title;
            this.Message = Message;
            Run();
        }
```

TrayBalloonFrm.cs

```
sing System;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Drawing2D;
namespace TrayBalloon
{
    public partial class TrayBalloonFrm : Form
    {
        public TrayBalloonFrm()
        {
            InitializeComponent();
            Title = null;
            StartingOffsetIndex = 0;
            SetStyle(ControlStyles.Selectable, false);
            MessageLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            MessageLabel.ForeColor = ForeColor = System.Drawing.Color.Red;
            MessageLabel.MouseDown += new MouseEventHandler(TrayBalloonFrm_MouseDown);
        }
        public string Title;
        public string Message;
        public volatile int StartingOffsetIndex;
        private float OpacityStep;
        public string SoundLocation;
        public string BackgroundLocation;
        private bool _LightWeight;
        public bool LightWeight
        {
            get
            {
                return _LightWeight;
            }
            set
            {
                _LightWeight = value;
            }
        }
         private Image _BackgroundImage;
        public override Image BackgroundImage
        {
            get
            {
                if (LightWeight)
                    return null;

                if (_BackgroundImage != null)
                    return _BackgroundImage;

                if (!string.IsNullOrEmpty(BackgroundLocation))
                {
                    try
                    {
                        _BackgroundImage = Bitmap.FromFile(BackgroundLocation);
                    }catch (System.IO.IOException)
                    { }
                    catch (ArgumentException)
                    { }
                    catch (UnauthorizedAccessException)
                    { }
                    catch (OutOfMemoryException)
                    { }
                }
                return base.BackgroundImage;
            }
            set
            {
                base.BackgroundImage = value;
            }
        }
        protected override void OnClosed(EventArgs e)
        {
            base.OnClosed(e);

            CloseTimer.Stop();
            MoveTimer.Stop();
        }
```

```
private double _OpacityValue;
        public double OpacityValue
        {
            get
            {
                if (UseOpacity)
                {
                    return Opacity;
                }
                else
                {
                    if (_OpacityValue < 0)
                        return 0;
                    else if (_OpacityValue > 1)
                        return 1;

                    return _OpacityValue;
                }
            }
            set
            {
                _OpacityValue = value;
                if (UseOpacity)
                    Opacity = value;
            }
             }

public bool UseOpacity = true;

        private void CloseTimer_Tick(object sender, EventArgs e)
        {
            if (Bounds.Contains(Cursor.Position))
                return;

            CloseTimer.Interval = MoveTimer.Interval / 2;

            if (OpacityValue == 0)
                     Close();


             else
                OpacityValue -= OpacityStep;

            MessageLabel.Refresh();
        }

        private void MoveTimer_Tick(object sender, EventArgs e)
        {
            OpacityValue += OpacityStep;
            if (Location.Y > Screen.PrimaryScreen.WorkingArea.Height - Height)
            {
                Location = new Point(Location.X, Location.Y - 2);
            }
            else
            {
                if (OpacityValue == 1.0)
                {
                    MoveTimer.Stop();
                    CloseTimer.Start();
                }
            }
        }
        private static readonly System.Text.RegularExpressions.Regex A = new System.Text.RegularExpressions.Regex(
            "\\<a\\W+href=\"(?<href>[^\"]*)\"\\W*\\>(?<text>[^\\<]*)\\</a\\>",
System.Text.RegularExpressions.RegexOptions.IgnoreCase | System.Text.RegularExpressions.RegexOptions.Multiline);
        private void SetupText()
        {
            TitleLabel.Text = Title;
            TitleLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            TitleLabel.ForeColor = ForeColor = System.Drawing.Color.Blue;
            string msg = Message ?? string.Empty;

            var matches = A.Matches(msg);
            if (matches == null || matches.Count == 0)
            {
                MessageLabel.Text = msg;
                MessageLabel.LinkArea = new LinkArea(msg.Length, 0);
            }
```

```
            else
            {
                StringBuilder sb = new StringBuilder();
                int last_index = 0;
                foreach (System.Text.RegularExpressions.Match match in matches)
                {
                    var href = match.Groups["href"].Value;
                    var text = match.Groups["text"].Value;

                    sb.Append(msg, last_index, match.Index - last_index);
                    MessageLabel.Links.Add(new LinkLabel.Link(sb.Length, text.Length) { LinkData = href });
                    sb.Append(text);
                    last_index = match.Index + match.Length;
                }
                if (last_index < msg.Length)
                    sb.Append(msg.Substring(last_index));
                MessageLabel.Text = sb.ToString();
            }
             }

        protected override void OnLoad(EventArgs e)
        {
            base.OnLoad(e);

            if (!UseOpacity)
                Opacity = 1;

            SetupText();

            Width = 200;
            Height = 60;
            Location = new Point(
                Screen.PrimaryScreen.Bounds.Width - Width,
                Screen.PrimaryScreen.Bounds.Height - Height * (1 + StartingOffsetIndex) + ((StartingOffsetIndex == 0) ? 0
: (Screen.PrimaryScreen.WorkingArea.Height - Screen.PrimaryScreen.Bounds.Height)));

            OpacityStep = (float)(((float)SystemInformation.WorkingArea.Height /
(float)SystemInformation.VirtualScreen.Height) / 10.0);

            MoveTimer.Start();

            Play(SoundLocation);
        }

        private static void Play(string SoundLocation)
        {
            if (string.IsNullOrEmpty(SoundLocation))
                return;

            try
            {
                System.Media.SoundPlayer Player = new System.Media.SoundPlayer();
                Player.SoundLocation = SoundLocation;
                Player.Play();
            }
            catch (System.IO.IOException)
            { }
            catch (UnauthorizedAccessException)
            { }
            catch (InvalidOperationException)
            { }
            catch (System.ServiceProcess.TimeoutException)
            { }
             }

        private void TrayBalloonFrm_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == System.Windows.Forms.MouseButtons.Right)
                Close();
        }
        protected override void OnPaint(PaintEventArgs e)
        {
            base.OnPaint(e);

            if (LightWeight)
            {
                LinearGradientBrush lgb = new LinearGradientBrush(
                    ClientRectangle, SystemColors.MenuHighlight, Color.LightBlue, 90);
                e.Graphics.FillRectangle(lgb, ClientRectangle);
                e.Graphics.DrawRectangle(new Pen(Color.White, 3), ClientRectangle);
                e.Graphics.DrawRectangle(new Pen(SystemColors.MenuHighlight, 1), ClientRectangle);
             }
```

```
public const int WM_NCCALCSIZE = 0x83;
        protected override void WndProc(ref Message m)
        {
            if (m.Msg == WM_NCCALCSIZE)
            { return; }
            base.WndProc(ref m);
        }
        private void MessageLabel_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            if (e.Link == null)
                return;

            var Link = e.Link.LinkData as string;

            if (string.IsNullOrEmpty(Link))
                return;

            try
            {
                var psi = new System.Diagnostics.ProcessStartInfo();
                psi.UseShellExecute = true;
                psi.FileName = Link;
                System.Diagnostics.Process.Start(Link);
            }
            catch (System.IO.IOException)
            { }
            catch (InvalidOperationException)
            { }
            catch (ArgumentException)
            { }
            catch (System.ComponentModel.Win32Exception)
            { }
        }

}

}

}
```

ChangeDisplayName.cs

```
using System;
using System.Windows.Forms;
namespace Erasmus
{
    public partial class ChangeDisplayName : Form
    {
        public ChangeDisplayName()
        {
            InitializeComponent();
        }

        private void btnChange_Click(object sender, EventArgs e)
        {
            if (txtName.Text == "")
                MessageBox.Show("Empty text box", "Error!");
            else
            {
                Global.server.ChangeDisplayName(Global.username, txtName.Text);
                MessageBox.Show("Successful changed");
                this.Close();
            }
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void ChangeDisplayName_Load(object sender, EventArgs e)
        {
            lblusername.Text += Global.username;
        }
    }
     }
```

Document.cs

```
using System;
using System.Windows.Forms;
using System.Net;
using System.IO;
namespace Erasmus
{
    public partial class Document : Form
    {
        public string server = "ftp://192.168.56.1/Exchange_Document";
        public string user_name = "PHAM HOANG";
        public string user_pass = "01101999hoang";
        public Document()
        {
            InitializeComponent();
        }

        private void проАвторToolStripMenuItem_Click(object sender, EventArgs e)
        {
            About about = new About();
            about.Show();
        }

        private void link1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            try {

                string path = @"C:\Users\famsu\Desktop\Document_Erasmus\";
                string fileName = "Application_Form.docx";
                FtpWebRequest request = (FtpWebRequest)WebRequest.Create(server + "/" + fileName);
                request.Method = WebRequestMethods.Ftp.DownloadFile;
                request.Credentials = new NetworkCredential(user_name, user_pass);
                FtpWebResponse response = (FtpWebResponse)request.GetResponse();
                using (Stream responseStream = response.GetResponseStream())
                {
                    using (Stream fileStream = new FileStream(path + fileName, FileMode.CreateNew))
                    {
                        responseStream.CopyTo(fileStream);
                    }
                }
                MessageBox.Show("Successful download file", "Information", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
            catch (WebException ex)
            {
                throw new Exception((ex.Response as FtpWebResponse).StatusDescription);
            }
             }

        private void link2_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            try {

                string path = @"C:\Users\famsu\Desktop\Document_Erasmus\";
                string fileName = "Learning_Agreement.docx";
                FtpWebRequest request = (FtpWebRequest)WebRequest.Create(server + "/" + fileName);
                     request.Method = WebRequestMethods.Ftp.DownloadFile;

request.Credentials = new NetworkCredential(user_name, user_pass);
                     FtpWebResponse response = (FtpWebResponse)request.GetResponse();

            using (Stream responseStream = response.GetResponseStream())
                            {
                                using (Stream fileStream = new FileStream(path + fileName, FileMode.CreateNew))
                                {
                                    responseStream.CopyTo(fileStream);
                                }
                            }
                            MessageBox.Show("Successful download file", "Information", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
                        }
                        catch (WebException ex)
                        {
                            throw new Exception((ex.Response as FtpWebResponse).StatusDescription);
                        }
                    }

private void link3_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            try
            {

                string path = @"C:\Users\famsu\Desktop\Document_Erasmus\";
                string fileName = "application.jpg";
```

```
                FtpWebRequest request = (FtpWebRequest)WebRequest.Create(server + "/" + fileName);
                 request.Method = WebRequestMethods.Ftp.DownloadFile;
                 request.Credentials = new NetworkCredential(user_name, user_pass);
                 FtpWebResponse response = (FtpWebResponse)request.GetResponse();
                 using (Stream responseStream = response.GetResponseStream())
                 {
                     using (Stream fileStream = new FileStream(path + fileName, FileMode.CreateNew))
                     {
                         responseStream.CopyTo(fileStream);
                     }
                 }
                 MessageBox.Show("Successful download file", "Information", MessageBoxButtons.OK,
MessageBoxIcon.Information);
             }
             catch (WebException ex)
             {
                 throw new Exception((ex.Response as FtpWebResponse).StatusDescription);
             }
         }
         private void link4_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
         {
             try
             {
                 string path = @"C:\Users\famsu\Desktop\Document_Erasmus\";
                 string fileName = "IPN.docx";
                 FtpWebRequest request = (FtpWebRequest)WebRequest.Create(server + "/" + fileName);
                 request.Method = WebRequestMethods.Ftp.DownloadFile;
                 request.Credentials = new NetworkCredential(user_name, user_pass);

FtpWebResponse response = (FtpWebResponse)request.GetResponse();
                 using (Stream responseStream = response.GetResponseStream())
                 {using (Stream fileStream = new FileStream(path + fileName, FileMode.CreateNew))
                     {
                         responseStream.CopyTo(fileStream);
                     }
}

MessageBox.Show("Successful download file", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);
             }
             catch (WebException ex)
             {
                 throw new Exception((ex.Response as FtpWebResponse).StatusDescription);
             }
         }
     }
}
```

Form1.cs

```
using System;
using System.Collections;
using System.Windows.Forms;
using LanMessengerLib;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Services;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Http;
using System.IO;
using System.Runtime.Remoting.Channels.Tcp;
using LanMessengerChatRoomBase;
using System.Net;
namespace Erasmus
{
    public partial class Form1 : Form
    {
       HttpChannel channel;
       string hostIP;
       private string[] setting = new string[6];
        public Form1()
        {
            InitializeComponent();
            Application.EnableVisualStyles();
            InitializeComponent();
            if (File.Exists("UserSetting.dat"))
                ReadUserSetting();
            else
                setting[5] = "1";

            channel = new HttpChannel();
            ChannelServices.RegisterChannel(channel);
            statusBar.Text = "Setting...";
            if (File.Exists("NetSetting.Dat"))
            {
                FileStream fs = new FileStream("NetSetting.Dat", FileMode.Open);
```

```
                BinaryReader br = new BinaryReader(fs);
                hostIP = br.ReadString();
                fs.Close();
                br.Close();
            }
            else
            {
                hostIP = "127.0.0.1";
            }
            statusBar.Text = "Successful.";
            try
            {
                MarshalByRefObject obj = (MarshalByRefObject)RemotingServices.Connect(typeof(IServer), "http://" + hostIP
+ ":9090/Server");
                Global.server = obj as IServer;
                (obj as RemotingClientProxy).Timeout = 5000;
            }
            catch
            {
                return;
            }
             }

private void Form1_Load(object sender, EventArgs e)
        {
            this.Show();
            SignIn();
        }
        private void UpdatePanelContact()
        {
            statusBar.Text = "Download list friend from server...";
            ArrayList contacts = null;
            try
            {
                contacts = Global.server.GetContacts(Global.username);
            }
            catch
            {
                AbortConnection();
                return;
            }
            statusBar.Text = "Successful download.";
            Label lblInfo = new Label();
            lblInfo.Text = "List friend :";
            lblInfo.Location = new System.Drawing.Point(8, 4);
            lblInfo.Size = new System.Drawing.Size(200, 16);
            pnlContacts.Controls.Clear();
            Global.contactList.Clear();
            pnlContacts.Controls.Add(lblInfo);
            int i = 20;
            statusBar.Text = "Update list friend...";
            foreach (object o in contacts)
            {
                LanMessengerControls.LanMessengerContact temp = new LanMessengerControls.LanMessengerContact();
                temp.DisplayName = Global.server.GetfullName(o as string);
                temp.Contact = o as string;
                statusBar.Text = "Adding " + o as string;
                try
                {
                    temp.Online = Global.server.IsVisible(o as String);
                }
                catch
                {
                    AbortConnection();
                    return;
                }
                temp.Location = new System.Drawing.Point(8, i);
                temp.Size = new System.Drawing.Size(pnlContacts.Width - 32, 16);
                temp.DoubleClick += new System.EventHandler(this.Contact_Click);
                temp.ContextMenu = conMenu;

                pnlContacts.Controls.Add(temp);
                Global.contactList.Add(temp);
                i += 18;
            }
            CreateAutoCompleteTextBox();
            statusBar.Text = "Update successful.";
             }

        private void SignIn()
        {
            notifyIcon.ContextMenu = null;
            statusBar.Text = "Sign in";
                 lblWelcome.Text = "Not sign in";
```

```
            FormSignIN frmSignIn = new FormSignIN();
            switch (frmSignIn.ShowDialog(this))
            {
                case DialogResult.OK:
                    Global.username = frmSignIn.txtUsername.Text;
                    try
                    {
                        ArrayList offline = Global.server.ReceiveOffline(Global.username);
                        if (offline.Count > 0)
                        {
                            FormOfflineMessage frmOffline = new FormOfflineMessage(offline);
                            frmOffline.Show();
                        }
                    }
                    catch
                    {
                        AbortConnection();
                        return;
                    }

                    txtSearchName.Enabled = true;
                    rbtnOnline.Enabled = true;
                    rbtnInvisible.Enabled = true;

                    if (Global.server.IsVisible(Global.username))
                    {
                        rbtnOnline.Checked = true;
                    }
                    else
                        rbtnInvisible.Checked = true;

                    statusBar.Text = "Signed in";
                    if (Global.server.IsVisible(Global.username))
                        lblWelcome.Text = "Hello " + Global.server.GetfullName(Global.username) + "! you are online.";
                    else
                        lblWelcome.Text = "Hello " + Global.server.GetfullName(Global.username) + "! you are invisible.";
                    tmrMessageReceive.Enabled = true;
                    tmrContactUpdate.Enabled = true;

                    this.menuAddContact.Enabled = true;
                    this.menuSendMessage.Enabled = true;
                    this.menuRemoveContact.Enabled = true;
                    this.menuLogMessage.Enabled = true;
                    this.menuChangeDisplayName.Enabled = true;
                    this.menuOpenChatRoom.Enabled = true;
                    this.menuJoinRoom.Enabled = true;
                    this.menuChangeUser.Text = "Sign in with another account...";
                    this.menuSignOut.Enabled = true;
                    this.pnlContacts.ContextMenu = this.conMenuContactsPanel;

                    this.notMenuSend.Enabled = true;
                    this.notMenuSignIn.Text = "Sign in with another account...";
                    this.notMenuSignOut.Enabled = true;

                    this.UpdatePanelContact();
                    break;
            }
            notifyIcon.ContextMenu = this.notifyMenu;
             }
        private void AbortConnection()
        {
            tmrMessageReceive.Enabled = false;
            tmrContactUpdate.Enabled = false;

            this.menuAddContact.Enabled = false;
            this.menuSendMessage.Enabled = false;
            this.menuRemoveContact.Enabled = false;
            this.menuLogMessage.Enabled = false;
            this.menuChangeDisplayName.Enabled = false;
            this.menuOpenChatRoom.Enabled = false;
            this.menuJoinRoom.Enabled = false;
            this.menuChangeUser.Text = "Sign in...";
            this.menuSignOut.Enabled = false;
            this.pnlContacts.ContextMenu = null;

            this.notMenuSend.Enabled = false;
            this.notMenuSignIn.Text = "Sign in...";
            this.notMenuSignOut.Enabled = false;
            ArrayList a = new ArrayList();
            foreach (string key in Global.windowList.Keys)
                a.Add(key);
            foreach (string key in a)
                (Global.windowList[key] as FormMessage).Close();
```

```
                Global.username = "";
                pnlContacts.Controls.Clear();
                Global.contactList.Clear();
                this.Focus();
                 }

        private void SignOut()
           {
                ArrayList a = new ArrayList();
                foreach (string key in Global.windowList.Keys)
                    a.Add(key);
                foreach (string key in a)
                    (Global.windowList[key] as FormMessage).Close();

                this.txtSearchName.Enabled = false;
                rbtnOnline.Enabled = false;
                rbtnInvisible.Enabled = false;
                this.txtSearchName.Clear();
                this.menuAddContact.Enabled = false;
                this.menuSendMessage.Enabled = false;
                this.menuRemoveContact.Enabled = false;
                this.menuLogMessage.Enabled = false;
                this.menuChangeDisplayName.Enabled = false;
                this.menuOpenChatRoom.Enabled = false;
                this.menuJoinRoom.Enabled = false;
                  this.menuChangeUser.Text = "Sign in...";


                  this.menuSignOut.Enabled = false;
                this.pnlContacts.ContextMenu = null;
                this.notMenuSend.Enabled = false;
                this.notMenuSignIn.Text = "Sign in...";
                this.notMenuSignOut.Enabled = false;
                tmrMessageReceive.Enabled = false;
                tmrContactUpdate.Enabled = false;
                if (setting[5] == "2")
                    DeleteAllLogs();

    try
                {
                    Global.server.SignOut(Global.username);
                    statusBar.Text = "Signed out.";
                }
                catch
                {
                    AbortConnection();
                    return;
                }
                finally
                {
                    pnlContacts.Controls.Clear();
                    Global.contactList.Clear();
                    Global.username = "";

                }
            }
            private void OpenFormMessage(string contact)
            {
                if (Global.windowList.Contains(contact))
                {
                    ((FormMessage)Global.windowList[contact]).Focus();
                }
                else
                {
                    FormMessage frmMessage = new FormMessage();
                    frmMessage.contact = contact;
                    frmMessage.Text = Global.server.GetfullName(contact) + " (" + contact + ")" + " - online message.";
                    frmMessage.Show();
                    Global.windowList.Add(contact, frmMessage);
                }
            }
            private void Form1_FormClosing(object sender, FormClosingEventArgs e)
            {
                notifyIcon.Dispose();
                ChannelServices.UnregisterChannel(channel);
                try
                {
                    if (setting[5] == "2")
                        DeleteAllLogs();
                    Global.server.SignOut(Global.username);
                }
                finally
                {
                    Application.Exit();
                }
                 }
```

```
 private void menuSendMessage_Click(object sender, EventArgs e)
        {
            using (FormSelectContact frmContact = new FormSelectContact())
            {
                if (frmContact.ShowDialog(this) == DialogResult.OK)
                {
                    if (frmContact.txtContact.Text == Global.username)
                    {
                        MessageBox.Show("You can't send yourself!");
                        return;
                    }
                     if (Global.windowList.Contains(frmContact.txtContact.Text))
                    {
                        ((FormMessage)Global.windowList[frmContact.txtContact.Text]).Focus();
                    }
                    else
                    {
                        FormMessage frmMessage = new FormMessage();
                        frmMessage.contact = frmContact.txtContact.Text;
                        frmMessage.Text = frmContact.txtContact.Text + " - online message.";
                        Global.windowList.Add(frmContact.txtContact.Text, frmMessage);
                        frmMessage.Show();
                    }
                }
            }
        }
        private void Contact_Click(object sender, System.EventArgs e)
        {
            OpenFormMessage(((LanMessengerControls.LanMessengerContact)sender).Contact);
        }
        private void tmrMessageReceive_Tick(object sender, EventArgs e)
        {
            LetterReceive letter;
            try
            {
                letter = Global.server.Receive(Global.username);
            }
            catch
            {
                AbortConnection();
                return;
            }
            if (letter.From == "")
            {
                return;
            }
            if (Global.windowList.Contains(letter.From))
            {
                if (letter.Message == "BUZZ IT")
                {
                    ((FormMessage)Global.windowList[letter.From]).Focus();
                }
                ((FormMessage)Global.windowList[letter.From]).AddText(letter.From, letter.Message);

            }
            else
            {
                FormMessage frmMessage = new FormMessage();
                frmMessage.contact = letter.From;
                frmMessage.Text = letter.From + " - online messageS.";
                frmMessage.AddText(letter.From, letter.Message);
                Global.windowList.Add(letter.From, frmMessage);
                frmMessage.Show();
            }
             }

    private void menuAddContact_Click(object sender, EventArgs e)
        {
            FormAddContact frmAddContact = new FormAddContact();
                 if (frmAddContact.ShowDialog() == DialogResult.OK)

{
                UpdatePanelContact();
            }
        }

        private void menuLogMessage_Click(object sender, EventArgs e)
        {
            FormLogReader f = new FormLogReader();
            f.Show();
        }
```

```
        private void menuRemoveContact_Click(object sender, EventArgs e)
        {
            FormSelectContact frmSelectContact = new FormSelectContact();
            if (frmSelectContact.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    Global.server.RemoveContact(Global.username, frmSelectContact.txtContact.Text);
                    UpdatePanelContact();
                }
                catch
                {
                    AbortConnection();
                    return;
                }
            }
             }

        private void menuChangeUser_Click(object sender, EventArgs e)
        {
            try
            {
                Global.server.SignOut(Global.username);
                if (setting[5] == "2")
                    DeleteAllLogs();
                pnlContacts.Controls.Clear();
                Global.contactList.Clear();
                this.txtSearchName.Enabled = false;
                rbtnOnline.Enabled = false;
                rbtnInvisible.Enabled = false;
                this.txtSearchName.Clear();
                this.menuAddContact.Enabled = false;
                this.menuSendMessage.Enabled = false;
                this.menuRemoveContact.Enabled = false;
                this.menuLogMessage.Enabled = false;
                this.menuChangeDisplayName.Enabled = false;
                  this.menuOpenChatRoom.Enabled = false;

                     this.menuJoinRoom.Enabled = false;
                this.menuChangeUser.Text = "Sign in...";
                this.menuSignOut.Enabled = false;
                this.pnlContacts.ContextMenu = null;
                this.notMenuSend.Enabled = false;
                  this.notMenuSignIn.Text = "Sign in...";

                     this.notMenuSignOut.Enabled = false;
                tmrMessageReceive.Enabled = false;
                tmrContactUpdate.Enabled = false;
                 }

    finally
            {
                SignIn();
            }
        }

        private void menuSignOut_Click(object sender, EventArgs e)
        {
            Global.server.SignOut(Global.username);
            if (setting[5] == "2")
                DeleteAllLogs();
            pnlContacts.Controls.Clear();
            Global.contactList.Clear();
            Global.username = "";
            this.txtSearchName.Enabled = false;
            rbtnOnline.Enabled = false;
            rbtnInvisible.Enabled = false;
            this.txtSearchName.Clear();
            this.menuAddContact.Enabled = false;
            this.menuSendMessage.Enabled = false;
            this.menuRemoveContact.Enabled = false;
            this.menuLogMessage.Enabled = false;
            this.menuChangeDisplayName.Enabled = false;
            this.menuOpenChatRoom.Enabled = false;
            this.menuJoinRoom.Enabled = false;
            this.menuChangeUser.Text = "Sign in...";
            this.menuSignOut.Enabled = false;
            this.pnlContacts.ContextMenu = null;
            this.notMenuSend.Enabled = false;
            this.notMenuSignIn.Text = "Sign in...";
            this.notMenuSignOut.Enabled = false;
            tmrMessageReceive.Enabled = false;
            tmrContactUpdate.Enabled = false;
            SignIn();
             }
```

```
          private void menuMin_Click(object sender, EventArgs e)
          {
          this.Hide();
             notifyIcon.BalloonTipTitle = "LAN MESSENGER!";
             notifyIcon.BalloonTipText = "Minimized.";
             notifyIcon.ShowBalloonTip(2000);
             this.notMenuMin.Enabled = false;
              }

          private void menuExit_Click(object sender, EventArgs e)
         {
             try
             {
                 if (setting[5] == "2")
                     DeleteAllLogs();
                 Global.server.SignOut(Global.username);
             }
             finally
             {
                 Application.Exit();
             }
              }

          private void DeleteAllLogs()
          {
            if (Global.username != "")
             {
                 string path = "logs/" + Global.username + "/";
                 if (Directory.Exists(path))
                 {
                     string[] directoryContact = Directory.GetDirectories(path);
                     foreach (string dc in directoryContact)
                     {
                         string[] fileLogs = Directory.GetFiles(dc);
                         foreach (string fl in fileLogs)
                         {
                             File.Delete(fl);
                         }
                         Directory.Delete(dc);
                     }
                     Directory.Delete(path);
                 }
             }
         }
         private void CreateAutoCompleteTextBox()
         {
             string[] arrayContact = new string[Global.contactList.Count];
             int i = 0;
             foreach (object o in Global.contactList)
             {
                 arrayContact[i] = ((LanMessengerControls.LanMessengerContact)o).Contact.ToString();
                 i++;
                  }

AutoCompleteStringCollection auto = new AutoCompleteStringCollection();
             foreach (string name in arrayContact)
             {
                 auto.Add(name + " (" + Global.server.GetfullName(name) + ")");
             }
             txtSearchName.AutoCompleteCustomSource = auto;
         }
         public void ReadUserSetting()
         {
             FileStream fs1 = new FileStream("UserSetting.dat", FileMode.Open);
             BinaryReader w1 = new BinaryReader(fs1);
             setting[0] = w1.ReadString().ToString();
             setting[1] = w1.ReadString().ToString();
             setting[2] = w1.ReadString().ToString();
             setting[3] = w1.ReadString().ToString();
             setting[4] = w1.ReadString().ToString();
             setting[5] = w1.ReadString().ToString();
             w1.Close();
             fs1.Close();
              }
```

```
         private void menuItem11_Click(object sender, EventArgs e)
        {
            FormOption fo = new FormOption();
            fo.Show();
             }

         private void menuChangeDisplayName_Click(object sender, EventArgs e)
        {
            ChangeDisplayName fcdn = new ChangeDisplayName();
            fcdn.Show();
        }
        private void menuNetworkSettings_Click(object sender, EventArgs e)
        {
            FormNetworkSetting frmNetworkSettings = new FormNetworkSetting();
            frmNetworkSettings.txtIP.Text = hostIP;
            if (frmNetworkSettings.ShowDialog() == DialogResult.OK)
            {
                hostIP = frmNetworkSettings.txtIP.Text;
                try
                {
                    AbortConnection();
                    MarshalByRefObject obj = (MarshalByRefObject)RemotingServices.Connect(typeof(IServer), "http://" +
hostIP + ":9090/Server");
                    Global.server = obj as IServer;
                }
                catch
                {
                    AbortConnection();
                    return;
                }
            }
             }

        private void GoOnline(string ContactName)
        {
            TrayBalloon.TrayBalloon tb = new TrayBalloon.TrayBalloon();
            tb.BackgroundLocation = "background.bmp";
            if (File.Exists(setting[1]))
                tb.SoundLocation = setting[1];
            else
                tb.SoundLocation = "sounds/Online.wav";
            tb.Title = "Lan Messenger!";
            tb.Message = Global.server.GetfullName(ContactName) + " đã Online!";
            tb.Run();
        }
        bool Check = false;
        private void GoOffline(string ContactName)
        {
            TrayBalloon.TrayBalloon tb = new TrayBalloon.TrayBalloon();
            tb.BackgroundLocation = "background.bmp";
            if (File.Exists(setting[2]))
                tb.SoundLocation = setting[2];
            else
                tb.SoundLocation = "sounds/Offline.wav";
            tb.Title = "Lan Messenger!";
            tb.Message = Global.server.GetfullName(ContactName) + " Offline!";
            tb.Run();
             }

        private void menuMusicOnline_Click(object sender, EventArgs e)
        {
            FormMusicPlayer fmp = new FormMusicPlayer();
            fmp.Show();
             }

     bool ChatRoomClosed = false;
        public void GetValue(Boolean b)
        {
            ChatRoomClosed = b;
            if (ChatRoomClosed)
                ChannelServices.UnregisterChannel(channelChatRoom);
             }

        private void menuOpenChatRoom_Click(object sender, EventArgs e)
        {
            channelChatRoom = new TcpChannel(7070);
            ChannelServices.RegisterChannel(channelChatRoom, false);
            RemotingConfiguration.RegisterWellKnownServiceType(typeof(SampleObject), Global.username.ToString(),
WellKnownObjectMode.Singleton);
```

```
if (OpenRoom() == 0)
            {
                ChannelServices.UnregisterChannel(channelChatRoom);
                channelChatRoom = null;
            }
        }
        TcpChannel chan;
        private int OpenRoom()
        {
            ArrayList alOnlineUser = new ArrayList();
            FormChatRoom objChatWin;

            if (true)
            {
                chan = new TcpChannel();
                chan = null;
                IPHostEntry temp = Dns.GetHostByName(Dns.GetHostName().ToString());
                string IP = temp.AddressList[0].ToString();
                objChatWin = new FormChatRoom();
                objChatWin.MyGetData = new FormChatRoom.GetString(GetValue);
                objChatWin.remoteObj = (SampleObject)Activator.GetObject(typeof(LanMessengerChatRoomBase.SampleObject),
"tcp://" + IP + ":7070/" + Global.username);

                if (!objChatWin.remoteObj.JoinToChatRoom(Global.username))
                {
                    MessageBox.Show(Global.username + " Signed in!");
                    ChannelServices.UnregisterChannel(chan);
                    chan = null;
                    objChatWin.Dispose();
                    return 1;
                }
                objChatWin.key = objChatWin.remoteObj.CurrentKeyNo();
                objChatWin.yourName = Global.username;
                objChatWin.Show();
                return 2;
            }
            else
            {
                MessageBox.Show("Error");
                chan = null;
                return 0;
            }
        }
        private void menuJoinRoom_Click(object sender, EventArgs e)
        {
            FormJoinRoom fjr = new FormJoinRoom();
            fjr.Show();
             }


         TcpChannel channelChatRoom;
        private void menuItem14_Click(object sender, EventArgs e)
        {
            if (File.Exists("help.chm"))
                System.Diagnostics.Process.Start("help.chm");
            else
                MessageBox.Show("Cant find help.chm", "Error");
               }


private void menuAbout_Click(object sender, EventArgs e)
        {
            About about = new About();
            about.Show();
        }

        private void tmrContactUpdate_Tick(object sender, EventArgs e)
        {
            try
            {
                foreach (object o in Global.contactList)
                {
                    if (((LanMessengerControls.LanMessengerContact)o).Online != Global.server.IsVisible((o as
LanMessengerControls.LanMessengerContact).Contact)) Check = !Check;
                    ((LanMessengerControls.LanMessengerContact)o).Online = Global.server.IsVisible((o as
LanMessengerControls.LanMessengerContact).Contact);
                    if (Check)
                    {
if (Global.server.IsVisible((o as LanMessengerControls.LanMessengerContact).Contact))
                            GoOnline((o as LanMessengerControls.LanMessengerContact).Contact.ToString());
                        else GoOffline((o as LanMessengerControls.LanMessengerContact).Contact.ToString());
                        Check = !Check;
                    }}
                pnlContacts.Update();
            }
```

```
            catch
            {
                AbortConnection();
                return;
            }
        }

        private void Form1_Resize(object sender, EventArgs e)
        {
            pnlContacts.Width = this.Width - 24;
            pnlContacts.Height = this.Height - 88;
        }

        private void pnlContacts_Resize(object sender, EventArgs e)
        {
            foreach (object o in Global.contactList)
            {
                ((LanMessengerControls.LanMessengerContact)o).Width = pnlContacts.Width - 32;
            }
        }

        private void rbtnOnline_CheckedChanged(object sender, EventArgs e)
        {
            if (rbtnOnline.Checked == true && !Global.server.IsVisible(Global.username))
            {
                Global.server.ChangeStatus(Global.username);
                lblWelcome.Text = "Hello " + Global.server.GetfullName(Global.username) + "! You are Online.";
            }
        }

        private void rbtnInvisible_CheckedChanged(object sender, EventArgs e)
        {
            if (rbtnInvisible.Checked == true && Global.server.IsVisible(Global.username))
            {
                Global.server.ChangeStatus(Global.username);
                lblWelcome.Text = "Hello " + Global.server.GetfullName(Global.username) + "! You are invisible.";
            }
        }
        private void picSearch_Click(object sender, EventArgs e)
        {
            if (txtSearchName.Text != "" && txtSearchName.Text != "Enter friend name...")
            {
                OpenFormMessage(txtSearchName.Text.Substring(0, txtSearchName.Text.IndexOf('(') - 1));
            }
        }
        private void notifyIcon_DoubleClick(object sender, EventArgs e)
        {
            this.Show();
            this.WindowState = FormWindowState.Normal;
        }
        private void txtSearchName_Click(object sender, EventArgs e)
        {
            this.txtSearchName.Clear();
            this.txtSearchName.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.txtSearchName.ForeColor = System.Drawing.SystemColors.MenuText;
        }
        private void txtSearchName_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                if (txtSearchName.Text.IndexOf('(') != -1)
                    OpenFormMessage(txtSearchName.Text.Substring(0, txtSearchName.Text.IndexOf('(') - 1));
                this.txtSearchName.Clear();
                this.txtSearchName.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                this.txtSearchName.ForeColor = System.Drawing.SystemColors.WindowFrame;
                this.txtSearchName.Text = "Enter friend name...";
            }
        }
    }
class Global
    {
        internal static IServer server;
        internal static Hashtable windowList;
        internal static ArrayList contactList;
        internal static string username;
        static Global()
        {
            windowList = new Hashtable();
            contactList = new ArrayList();
            username = "";
        }
    }
     }
```

FormAddContact.cs

```
using System;
using System.Windows.Forms;
namespace Erasmus
{
    public partial class FormAddContact : Form
    {
        private System.Windows.Forms.Label label1_1;
        private System.Windows.Forms.TextBox txtContact_1;
        private System.Windows.Forms.Button btnAdd_1;
        private System.Windows.Forms.Button btnCancel_1;
        private System.ComponentModel.Container components_1 = null;
        public FormAddContact()
        {
            InitializeComponent();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            if (Global.username == txtContact.Text)
            {
                MessageBox.Show("You cant add yourself to list!");
                txtContact.Text = "";
                this.DialogResult = DialogResult.None;
                return;
            }
            if (Global.server.AddContact(Global.username, txtContact.Text))
            {
                return;
            }
            else
            {

                bool check = false;
                foreach (object o in Global.server.GetContacts(Global.username))
                    if (o.ToString() == txtContact.Text)
                    {
                        check = true;
                        break;
                    }
                if (check) MessageBox.Show("This user is already in your list");
                else
                    MessageBox.Show("This user is not existing!");
                this.DialogResult = DialogResult.None;
            }
        }

        private void FormAddContact_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (this.DialogResult == DialogResult.None)
                e.Cancel = true;
        }
    }
     }
```

FormChangePassword.cs

```
using System;
using System.Windows.Forms;
namespace Erasmus
{
    public partial class FormChangePassword : Form
    {
        public FormChangePassword()
        {
            InitializeComponent();
        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            if (this.txtNewPassword.Text == txtConfirmPassword.Text)
            {
                try
                {
                    if (Global.server.ChangePassword(txtUsername.Text, txtCurPassword.Text, txtNewPassword.Text))
                    {
                        MessageBox.Show("Successful changed!");
                    }
                    else
                    {
                        MessageBox.Show("Wrong. Check it again!");
                        this.DialogResult = DialogResult.None;
                    }
                }
```

```
                catch
                {
                    MessageBox.Show("Connect error. Check it again!");
                }
            }
            else
            {
                MessageBox.Show("Retype password is not matched!");
                this.DialogResult = DialogResult.None;
            }
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void FormChangePassword_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (this.DialogResult == DialogResult.None)
            {
                e.Cancel = true;
            }
        }
    }
}
```

FormChatRoom.cs

```
using System;
using System.Windows.Forms;
using System.Collections;
using LanMessengerChatRoomBase;
namespace Erasmus
{
    public partial class FormChatRoom : Form
    {
        internal SampleObject remoteObj;
        internal int key = 0;
        internal string yourName;
        public delegate void GetString(Boolean b);
        public GetString MyGetData;

        ArrayList alOnlineUser = new ArrayList();
        int skipCounter = 4;
        ArrayList onlineUser;
        public FormChatRoom()
        {
            InitializeComponent();
            foreach (object o in Global.contactList)
            {

cbbListContact.Items.Add(Global.server.GetfullName(((LanMessengerControls.LanMessengerContact)o).Contact.ToString()) + "
(" + ((LanMessengerControls.LanMessengerContact)o).Contact.ToString() + ")");
            }
        }
        private void btnSend_Click(object sender, EventArgs e)
        {
            SendMessage();
        }
        private void SendMessage()
        {

            if (remoteObj != null && txtChatHere.Text.Trim().Length > 0)
            {
                remoteObj.SendMsgToSvr(Global.server.GetfullName(yourName) + ": " + txtChatHere.Text);
                txtChatHere.Text = "";
            }
        }
        private void timer1_Tick(object sender, EventArgs e)
        {
            if (remoteObj != null)
            {
                string tempStr = remoteObj.GetMsgFromSvr(key);
                if (tempStr.Trim().Length > 0)
                {
                    key++;
                    txtAllChat.Text = txtAllChat.Text + "\n" + tempStr;
                }
                {
                    onlineUser = remoteObj.GetOnlineUser();
                    lstOnlineUser.DataSource = onlineUser;
                    skipCounter = 0;
```

```
                    if (onlineUser.Count < 2)
                    {
                        txtChatHere.Text = "You should hae 2 participants";
                        txtChatHere.Enabled = false;
                    }
                    else if (txtChatHere.Text == "You should hae 2 participants" && txtChatHere.Enabled == false)
                    {
                        txtChatHere.Text = "";
                        txtChatHere.Enabled = true;
                    }
                }
            }
        }
    }
}
```

FormFileReceive.cs

```
using System;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Diagnostics;
using System.Threading;
namespace Erasmus
{
    public partial class FormFileReceive : Form
    {
        private string[] setting = new string[5];
        Thread t1;
        int flag = 0;
        string receivedPath;
        public delegate void UpdateStatusDelegate(string newStatus);
        public FormFileReceive()
        {
            InitializeComponent();
            if (File.Exists("UserSetting.dat"))
                ReadUserSetting();

            t1 = new Thread(new ThreadStart(StartListening));
            t1.Start();
            if (Directory.Exists(setting[4]))
                receivedPath = setting[4];
            else
                receivedPath = Path.GetFullPath("files");
            lblPath.Text = receivedPath;
        }
        public void ReadUserSetting()
        {
            FileStream fs1 = new FileStream("UserSetting.dat", FileMode.Open);
            BinaryReader w1 = new BinaryReader(fs1);
            setting[0] = w1.ReadString().ToString();
            setting[1] = w1.ReadString().ToString();
            setting[2] = w1.ReadString().ToString();
            setting[3] = w1.ReadString().ToString();
            setting[4] = w1.ReadString().ToString(); //
            w1.Close();
            fs1.Close();
        }
        public class StateObject
        {
            // Client socket.
            public Socket workSocket = null;

            public const int BufferSize = 1024;
            // Receive buffer.
            public byte[] buffer = new byte[BufferSize];
             }

          public static ManualResetEvent allDone = new ManualResetEvent(false);
        public void StartListening()
        {
            byte[] bytes = new Byte[1024];
            IPEndPoint ipEnd = new IPEndPoint(IPAddress.Any, 6565);
            Socket listener = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
                try
```

```
                listener.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReuseAddress, true);
                listener.Bind(ipEnd);
                listener.Listen(100);
                while (true)
                {
                    allDone.Reset();
                    listener.BeginAccept(new AsyncCallback(AcceptCallback), listener);
                    allDone.WaitOne();
                    UpdateStatus("Successful received!");
                }
            }
            catch (Exception ex)
            {
                UpdateStatus(ex.Message);
            }
        }
        public void AcceptCallback(IAsyncResult ar)
        {
            allDone.Set();
            Socket listener = (Socket)ar.AsyncState;
            Socket handler = listener.EndAccept(ar);
            StateObject state = new StateObject();
            state.workSocket = handler;
            handler.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
            new AsyncCallback(ReadCallback), state);
            flag = 0;
        }
        public string FileName;
        public void ReadCallback(IAsyncResult ar)
        {
            int fileNameLen = 1;
            receivedPath = receivedPath + "\\";
            String content = String.Empty;
            StateObject state = (StateObject)ar.AsyncState;
            Socket handler = state.workSocket;
                 int bytesRead = handler.EndReceive(ar);

              if (bytesRead > 0)
            {
                if (flag == 0)
                {
                    fileNameLen = BitConverter.ToInt32(state.buffer, 0);
                    string fileName = Encoding.UTF8.GetString(state.buffer, 4, fileNameLen);
                    FileName = fileName;
                    UpdateStatus("Receiving file...");
                    flag++;
                }
                if (flag >= 1)
                {
                    BinaryWriter writer = new BinaryWriter(File.Open(receivedPath + FileName, FileMode.Append));
                    if (flag == 1)
                    {
                        writer.Write(state.buffer, 4 + fileNameLen, bytesRead - (4 + fileNameLen));
                        flag++;
                         }

else
                        writer.Write(state.buffer, 0, bytesRead);
                    writer.Close();
                    handler.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
                    new AsyncCallback(ReadCallback), state);
                }
            }
            else
            {
                UpdateStatus("Successful received!");
            }
        }
        private void btnOpenFolder_Click(object sender, EventArgs e)
        {
            Process ps = new Process();

            ps.StartInfo.FileName = Path.GetFullPath(receivedPath);
            ps.Start();
            ps.Close();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            if (lblStatus.Text != "Successful received!")
                MessageBox.Show("Please wait");
```

```
            else
            {
                this.Hide();
            }
        }
        private void UpdateStatus(string newStatus)
        {
            if (lblStatus.InvokeRequired)
            {
                try
                {
                    UpdateStatusDelegate del = new UpdateStatusDelegate(UpdateStatus);
                    lblStatus.Invoke(del, new object[] { newStatus });
                }
                catch (Exception ex)
                {
                    lblStatus.Text = ex.ToString();
                }
            }
            else
            {
                lblStatus.Text = newStatus;
            }
        }
        private void FormFileReceive_FormClosing(object sender, FormClosingEventArgs e)
        {
            e.Cancel = true;
            this.Hide();
        }
    }
}
```

FormJoinRoom.cs

```
using System;
using System.Windows.Forms;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using System.Collections;
using LanMessengerChatRoomBase;
namespace Erasmus
{
    public partial class FormJoinRoom : Form
    {
        public FormJoinRoom()
        {
            InitializeComponent();
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void btnJoin_Click(object sender, EventArgs e)
        {
            if (txtNick.Text != "")
            {
                if (Global.server.IsVisible(txtNick.Text))
                    OpenRoom();
                else
                    MessageBox.Show("User is not online", "Error!");
            }
            else
                MessageBox.Show("Nick of creator", "Error!");
        }
        bool ChatRoomClosed = false;
        public void GetValue(Boolean b)
        {
            ChatRoomClosed = b;
            if (ChatRoomClosed)
            {
                     ChannelServices.UnregisterChannel(chan);

                            chan = null;
            }
        }
       TcpChannel chan;
        private void OpenRoom()
        {
            ArrayList alOnlineUser = new ArrayList();
            FormChatRoom objChatWin;
```

```
            if (chan == null)
            {
                chan = new TcpChannel();
                     ChannelServices.RegisterChannel(chan, false);


                     objChatWin = new FormChatRoom();
                objChatWin.MyGetData = new FormChatRoom.GetString(GetValue);
                objChatWin.remoteObj = (SampleObject)Activator.GetObject(typeof(LanMessengerChatRoomBase.SampleObject),
"tcp://" + Global.server.GetIP(txtNick.Text) + ":7070/" + txtNick.Text);

                     if (!objChatWin.remoteObj.JoinToChatRoom(Global.username))

                    {
                      MessageBox.Show(Global.username + " sign in!");
                      ChannelServices.UnregisterChannel(chan);
                      chan = null;
                      objChatWin.Dispose();
                      return;
                }
                objChatWin.key = objChatWin.remoteObj.CurrentKeyNo();
                objChatWin.yourName = Global.username;

                this.Hide();
                objChatWin.Show();
            }
            else
            {
                MessageBox.Show("Error creating!");
                ChannelServices.UnregisterChannel(chan);
                chan = null;
            }
        }
    }
     }
```

FormLogReader.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.IO;
using System.Text.RegularExpressions;
namespace Erasmus
{
    public partial class FormLogReader : Form
    {
        public FormLogReader()
        {
            InitializeComponent();
            InitializeListView();
        }
        public void InitializeListView()
        {
            ColumnHeader header1 = this.lvListMessage.Columns.Add("Name", 27 *
Convert.ToInt32(lvListMessage.Font.SizeInPoints), HorizontalAlignment.Center);
            ColumnHeader header2 = this.lvListMessage.Columns.Add("Time", 20 *
Convert.ToInt32(lvListMessage.Font.SizeInPoints), HorizontalAlignment.Center);
        }
        string path = "logs/" + Global.username + "/";
        string[] listContact;
        string[] listFile;

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
private void FormLogReader_Load(object sender, EventArgs e)
        {
            if (Directory.Exists(path))
            {
                listContact = Directory.GetDirectories(path);
                List<string> items = new List<string>();
                for (int i = 0; i < listContact.Length; i++)
                {
                    items.Add(GetFolderContact(listContact[i]));
                }
                lbListContact.DataSource = items;
            }
            else
            {
                btnDelete.Enabled = false;
            }
             }
```

```
 private string GetFileContact(string path)
{
    string pattern = "[0-9]+.dat";
    Regex filename = new Regex(pattern);
    Match m = filename.Match(path);
    if (m.Success)
    {
        return m.Value;
    }
    return "";
}

private string GetFolderContact(string path)
{
    string name;
    int index = path.IndexOf("/", 5) + 1;
    name = path.Substring(index, path.Length - index);
    return name;
     }

private string FileNametoTime(string name)
{
    string s;
    s = name;
    s = s.Insert(2, "/");
    s = s.Insert(5, "/");
    s = s.Substring(0, s.Length - 4);
    return s;
}
private void lbListContact_SelectedValueChanged(object sender, EventArgs e)
{
    lvListMessage.Clear();
    InitializeListView();
    listFile = Directory.GetFiles(path + lbListContact.SelectedItem.ToString());
    for (int i = 0; i < listFile.Length; i++)
    {
        AddItemstoListView(lbListContact.SelectedItem.ToString(), FileNametoTime(GetFileContact(listFile[i])));
    }
    rtbLogMessage.Clear();
}
private void AddItemstoListView(string Contact, string time)
{
    int n = this.lvListMessage.Items.Count;
    this.lvListMessage.Items.Add(Contact);
    this.lvListMessage.Items[n].SubItems.Add(time);
}
private void lvListMessage_SelectedIndexChanged(object sender, EventArgs e)
{
    int index = 0;
    if (this.lvListMessage.SelectedItems.Count > 0)
        index = this.lvListMessage.SelectedIndices[0];
    FileStream fs = new FileStream(listFile[index], FileMode.Open);
    StreamReader sr = new StreamReader(fs);
    rtbLogMessage.Clear();
    rtbLogMessage.AppendText(sr.ReadToEnd());
    fs.Close();
     }

 private void lvListMessage_ItemChecked(object sender, ItemCheckedEventArgs e)
{
    if (e.Item.Checked == true || lvListMessage.CheckedItems.Count > 0)
        btnDelete.Enabled = true;
    else
        btnDelete.Enabled = false;
     }

private void btnDelete_Click(object sender, EventArgs e)
{
    switch (MessageBox.Show("Are you sure", "Delete log chat...", MessageBoxButtons.YesNo))
    {
        case DialogResult.Yes:
            for (int i = 0; i < lvListMessage.Items.Count; i++)
            {
                if (lvListMessage.Items[i].Checked)
                {
                    lvListMessage.Items[i].Remove();
                    File.Delete(listFile[i]);
                    listFile = Directory.GetFiles(path + lbListContact.SelectedItem.ToString());
                    rtbLogMessage.Clear();
                }
            }
```

```
                break;
                case DialogResult.No:
                    this.DialogResult = DialogResult.None;
                    break;
            }
             }

    }
}
```

FormNetworkSetting.cs

```
using System;
using System.Windows.Forms;
using System.IO;
namespace Erasmus
{
    public partial class FormNetworkSetting : Form
    {
        public FormNetworkSetting()
        {
            InitializeComponent();
        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            switch (MessageBox.Show("Are you sure?.", "Change setting?", MessageBoxButtons.YesNoCancel))
            {
                case DialogResult.Yes:
                    FileStream fs = new FileStream("NetSetting.Dat", FileMode.Create);
                    BinaryWriter bw = new BinaryWriter(fs);
                    bw.Write(txtIP.Text);
                    bw.Close();
                    fs.Close();
                    return;
                    break;
                case DialogResult.No:
                    this.DialogResult = DialogResult.None;
                    break;
                case DialogResult.Cancel:
                    this.DialogResult = DialogResult.Cancel;
                    break;
            }
        }

        private void FormNetworkSetting_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (this.DialogResult == DialogResult.None)
                e.Cancel = true;
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

FormSendingFile.cs

```
using System;
using System.Text;
using System.Windows.Forms;
using System.Net.Sockets;
using System.IO;
namespace Erasmus
{
    public partial class FormSendingFile : Form
    {
        public static string IP;
        string splitter = "'\\'";
        string fName;
        string[] split = null;
        byte[] clientData;
        FormMessage f;
        public FormSendingFile()
        {
            InitializeComponent();
        }
        public FormSendingFile(FormMessage form)
        {
            InitializeComponent();
            f = new FormMessage();
            f = form;
        }
```

```
        private void btnSend_Click(object sender, EventArgs e)
        {
            Socket clientSock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
            byte[] fileName = Encoding.UTF8.GetBytes(fName);
            byte[] fileData = File.ReadAllBytes(txtFileLink.Text);
            byte[] fileNameLen = BitConverter.GetBytes(fileName.Length);
            clientData = new byte[4 + fileName.Length + fileData.Length];
            fileNameLen.CopyTo(clientData, 0);
            fileName.CopyTo(clientData, 4);
            fileData.CopyTo(clientData, 4 + fileName.Length);
            lblStatus.Text = "Connecting to receiver...";
            clientSock.Connect(IP, 6565);
            lblStatus.Text = "Sending...";
            clientSock.Send(clientData);
            clientSock.Close();
            lblStatus.Text = "Sending successful: " + Path.GetFileName(txtFileLink.Text);
        }
        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void btnSelectFile_Click(object sender, EventArgs e)
        {
            char[] delimiter = splitter.ToCharArray();
            openFileDialog1.ShowDialog();
            txtFileLink.Text = openFileDialog1.FileName;
            split = txtFileLink.Text.Split(delimiter);
            int limit = split.Length;
            fName = split[limit - 1].ToString();
            if (txtFileLink.Text != null)
                btnSend.Enabled = true;
        }
        private void FormSendingFile_Load(object sender, EventArgs e)
        {
            IP = f.IPContact;
        }
    }
```

Professor.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Net;
using System.IO;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace Erasmus
{
    public partial class Professor : Form
    {
        string username;
        string role;
        string faculty_2;
        int id;
        public string server = "ftp://192.168.56.1/Application";
        SqlConnection connection = new SqlConnection(connectionString);
        public string user_name = "PHAM HOANG";
        public string user_pass = "01101999hoang";
        static string connectionString = "Data Source =.\\SQLEXPRESS;Initial Catalog = Public; Integrated Security =
True;";

        public Professor(int id, string username, string faculty, string role)
        {
            InitializeComponent();
            this.username = username;
            this.faculty_2 = faculty;
            this.role = role;
            this.id = id;
        }
        public void Loading()
        {
            string query = "select Users.username, Doument.application, Doument.inp from Doument inner join Users on
Doument.id_sudent = Users.id ";
            SqlCommand command1 = new SqlCommand(query, connection);
            connection.Open();
            DataTable dt1 = new DataTable();
            SqlDataAdapter da1 = new SqlDataAdapter(command1);
            da1.Fill(dt1);
            document_down.DataSource = dt1;
            connection.Close();
```

```
            string query1 = "select Users.username, Doument.application, Doument.inp from Doument inner join Users on
    Doument.id_sudent = Users.id ";
            SqlCommand command2 = new SqlCommand(query, connection);
            connection.Open();
            DataTable dt2 = new DataTable();
            SqlDataAdapter da2 = new SqlDataAdapter(command2);
            da2.Fill(dt1);
            document_upl.DataSource = dt2;
            connection.Close();
        }
         public void UploadFile()
        {
            string FullName1 = this.zayava.Text;
            string FullName2 = this.inp.Text;
            FileInfo toUpload1 = new FileInfo(FullName1);
            FileInfo toUpload2 = new FileInfo(FullName2);
            FtpWebRequest request1 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload1.Name);

            request1.Method = WebRequestMethods.Ftp.UploadFile;
            request1.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
            Stream stream1 = request1.GetRequestStream();
            FileStream fs1 = File.OpenRead(FullName1);
            byte[] buffer1 = new byte[1024];
            double total1 = (double)fs1.Length;
            int byteRead1 = 0;
            do
            {
                byteRead1 = fs1.Read(buffer1, 0, 1024);
                stream1.Write(buffer1, 0, byteRead1);
            } while (byteRead1 != 0);
            fs1.Close();
            stream1.Close();
            //================Upload Certificate================//
            FtpWebRequest request2 = (FtpWebRequest)WebRequest.Create(server + "/" + toUpload2.Name);
            request2.Method = WebRequestMethods.Ftp.UploadFile;
            request2.Credentials = new NetworkCredential("PHAM HOANG", "01101999hoang");
            Stream stream2 = request2.GetRequestStream();
            FileStream fs2 = File.OpenRead(FullName2);
            byte[] buffer2 = new byte[1024];
            double total2 = (double)fs2.Length;
            int byteRead2 = 0;
            do
            {
                byteRead2 = fs2.Read(buffer2, 0, 1024);
                stream2.Write(buffer2, 0, byteRead2);
            } while (byteRead2 != 0);
            fs2.Close();
            stream2.Close();
        }

        }
    }

            MessageBox.Show("Successful edit", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        private void Professor_Load(object sender, EventArgs e)
        {
            username_1.Text = username;
            role_1.Text = role;
            faculty_1.Text = faculty_2;
            Loading();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Form1 form1 = new Form1();
            form1.Show();
        }
        private void download_Click(object sender, EventArgs e)
        {
            List<int> list_ind = new List<int>() { 1,  2 };
            string path = @"C:\Users\famsu\Desktop\";
            string folder;

            for (int i = 1; i < document_down.RowCount; i++)
            {
                folder = document_down.Rows[i].Cells[0].Value.ToString();
                System.IO.Directory.CreateDirectory(path + folder);
                foreach (int j in list_ind)
                {
                    string fileName = document_down.Rows[i].Cells[j].Value.ToString();
                    Download(folder, fileName);
                }
            }
```

```
            MessageBox.Show("Successful download!", "Information", MessageBoxButtons.OK);
        }

        public void Download(string folder, string fileName)
        {
            try
            {
                FtpWebRequest request = (FtpWebRequest)WebRequest.Create(server + "/" + fileName);
                request.Method = WebRequestMethods.Ftp.DownloadFile;
                request.Credentials = new NetworkCredential(user_name, user_pass);
                FtpWebResponse response = (FtpWebResponse)request.GetResponse();
                string folder_1 = folder + @"\";
                string path = @"C:\Users\famsu\Desktop\" + folder_1;
                using (Stream responseStream = response.GetResponseStream())
                {
                    using (Stream fileStream = new FileStream(path + fileName, FileMode.CreateNew))

                            {
                        responseStream.CopyTo(fileStream);
                    }
                }
            }

            catch (WebException ex)
            {
                throw new Exception((ex.Response as FtpWebResponse).StatusDescription);
            }
        }
        private void zayava_browse_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Document files (*.doc, *.docx, *.txt, *pdf) | *.doc; *.docx; *.txt; *.pdf" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
        zayava.Text = filename;
                }
            }
        }
        private void inp_browse_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog() { Multiselect = false, ValidateNames = true,
Filter = "Document files (*.doc, *.docx, *.txt, *pdf) | *.doc; *.docx; *.txt; *.pdf" })
            {
                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filename = openFileDialog.FileName;
               inp.Text = filename;
                }
            }
        }
        private void upload_Click(object sender, EventArgs e)
        {
            UploadFile();
            Sending();
        }
    }
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації і управління

УЗГОДЖЕНО
Керівник проєкту
_____ *Тєлишева Т. О.*
(підпис) (вл. ім’я, прізвище)
“13” квітня 2020 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл. ім’я, прізвище)
“14” квітня 2020 р.

СИСТЕМА ПІДТРИМКИ НАВЧАННЯ СТУДЕНТІВ – УЧАСНИКІВ ЄВРОПЕЙСЬКИХ ПРОГРАМ МОБІЛЬНОСТІ
ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 6324.01.000 ТЗ*

на 8 сторінках

Київ – 2020 року

ЗМІСТ

1. ЗАГАЛЬНЕ ПОЛОЖЕННЯ

1.1. Повне найменування системи та її умовне позначення

Система підтримки навчання студентів - учасників європейських програм мобільності

1.2. Найменування організації-замовника та організацій-учасників робіт

Відділ академічної мобільності студентів КПІ ім. Ігоря Сікорського. Розробником систем є Фам Суан Хоанг

1.3. Перелік документів, на підставі яких створюється системи

Підставою для розробки системи є завдання на переддипломну практику

1.4. Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку робіт по створенню системи – 13.04.2020.

Плановий робіт завершення робіт по створенню системи – 01.06.2020.

2. ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1. Призначення системи

Розробити система, яка дає студентам можливість шукати університет-партнер відповідно до категорії студенту і дає можливість надіслати заяви або ІНП до координатору, відділу мобільності, завідуючого факультету/кафедри на підписання.

2.2. Цілі створення системи

Основна мета проекту- зменшити втрати часу в процесі прийняття участі в програмах академічної мобільності за рахунок використання он-лайн технології оформлення документів .

Для досягнення цілі необхідно вирішити такі задачі:

- аналізпроцесівоформленнядокументів;
- розробка структури та функціональної моделі он-лайн технології оформлення документів;
- розробити інформаційне та програмне забезпечення для технології ;
- провести тестування технології та експериментальне впровадження.

3. ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Основними об'єктами автоматиції є процеси порівняння змісту курсів, процес подача документів і процес оформлення документів. Для цього необхідно мати узгодження з чотирма сторонами: учасники мобільності, відділ мобільності, координатор з мобільності, завідувач кафедри/ декан.

Студент:

- вибір університета-партнера за своїми категоріями;
- заповнення заяви, ІНП;
- коректно заповнення особистих даних;
- здати готові документи.

Координатор, завідувач кафедри/ декан:

— підписати заяву і ІНП.

Відділ мобільності:

- прийняти документи;
- порахувати рейтинг для окремих студентів;
- створити таблиці переможців для окремих університетів.

3.1. ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.2. Вимоги до функціонувальних характеристик

Розроблена система повинен виконувати такі функції:

–функція для процесу подачі документи;

–функція обчислення рейтинг для окремих студентів;

–фцнкція оформлення документів.

3.3. Вимоги до надійності

Система повинен коректно здійснювати коректно виконати процес подачі документів. . Крім цього, інформація щодо облікових записів адміністратора та користувачів повинна бути конфіденційною, щоб уникнути потенційних проблем з авторизацією в системі.

3.4. Вимоги до складу і параметрів технічних засобів

Склад технічних засобів визначається наявністю ЕОМ, яка має задовольняти наступні вимоги:

- операційна система – Window 7 з пакетом оновлення SP 1;
- оперативна пам’ять – 2 ГБ;
- місткість жорсткого диску – до 50 ГБ;
- процессор – 1.8 ГГц;

4. СТАДІЇ І ЕТАПИ РОЗРОБКИ

1. Ознайомлення із постановкою завдання.
2. Огляд існуючих реалізацій.
3. Збір матеріалів для розробки.
4. Вибір мову програмування.
5. Створення бази даних для учасників, бази даних для проект кредитної мобільності.
6. Розробка програмного продукту.
7. Тестування програмного продукту.
8. Формування звітності.
9. Здача готового програмного продукту.

5. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Для контролю замовником програмного продукту необхідно проводити випробування на ЕОМ замовника та ЕОМ учасника. Приймання готової роботи повинно відбутися не пізніше 17 травня 2020.

5.1.1. Види випробувань

Випробувань складається з 2 видів:

- тестове випробування – тестувати програмний продукт на ЕОМ з готових вхідних даних та виковати деякі функції як оформлення документів та функція обчислення рейтингу;
- експериментальне випробування – тестування програмний продукт на ЕОМ замовника, щоб тестувати функція, яка відповідає за процесу подачі документів

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

Дата перевірки:
11.06.2020 01:36:43 EEST

Дата звіту:
12.06.2020 01:31:16 EEST

ID перевірки:
1003947939

Тип перевірки:
Doc vs Internet + Library

ID користувача:
77149

Назва документу: SuanFam_is63

ID файлу: 1003963102 Кількість сторінок: 103 Кількість слів: 12833 Кількість символів: 91957 Розмір файлу: 3.98 MB

10.6% Схожість

Найбільша схожість: 4.05% з джерело https://ela.kpi.ua/bitstream/123456789/30923/1/Braiko_bakalavr.pdf

8.45% Схожість з Інтернет джерелами 117 ..... Page 105

8.76% Текстові збіги по Бібліотеці акаунту 617 ..... Page 107

0.67% Цитат

Цитати 4 ..... Page 108

Вилучення переліку посилань вимкнено

0% Вилучень

Вилучений текст відсутній

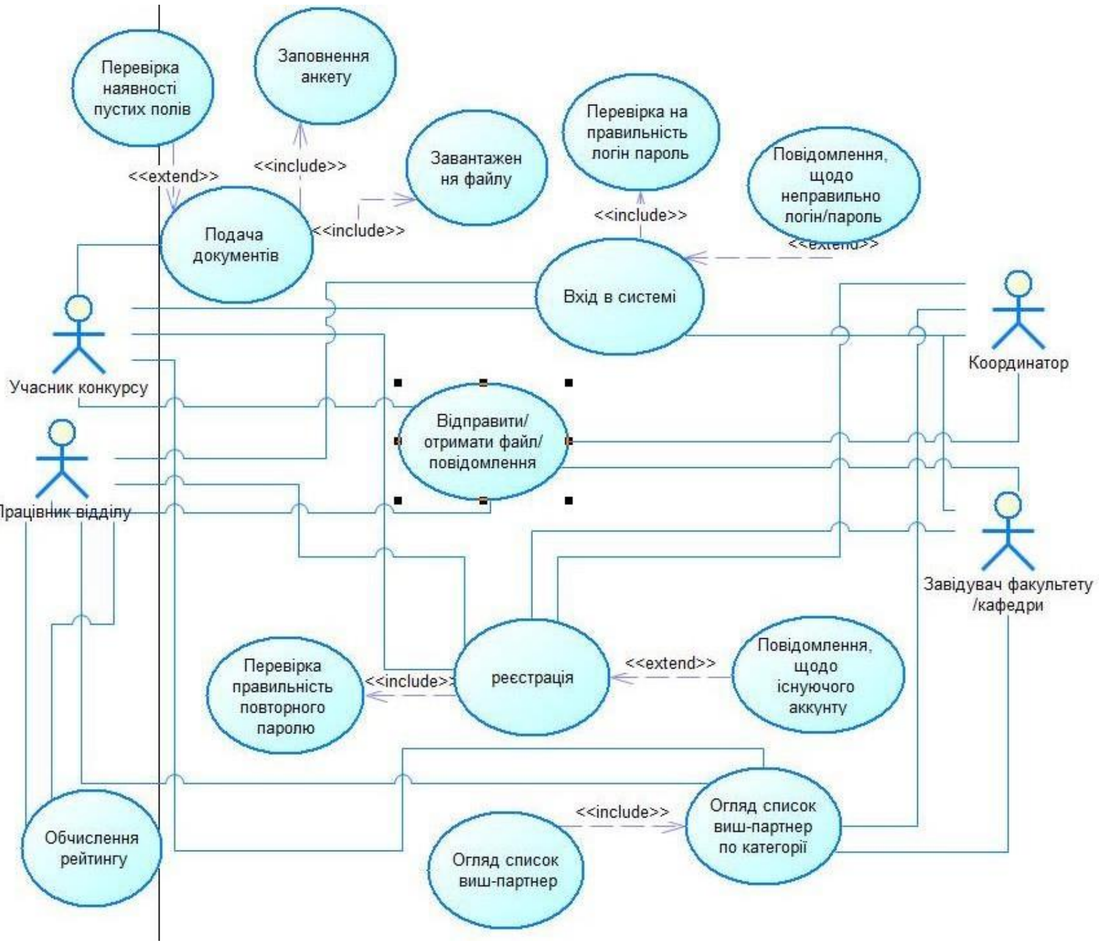
Підміна символів

Заміна символів 120

Графічний матеріал
до дипломного проєкту

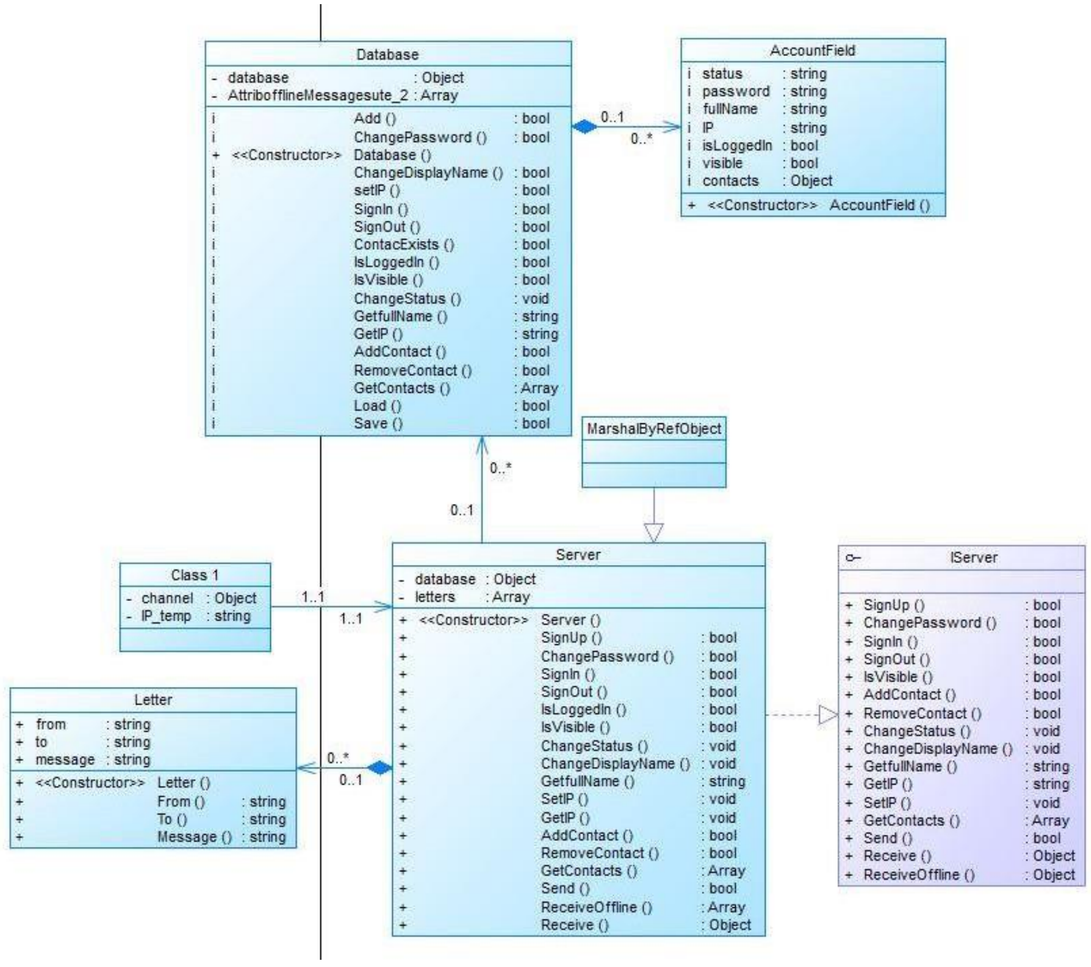
на тему: Система підтримки навчання студентів – учасників європейських програм мобільності.

Київ – 2020 року

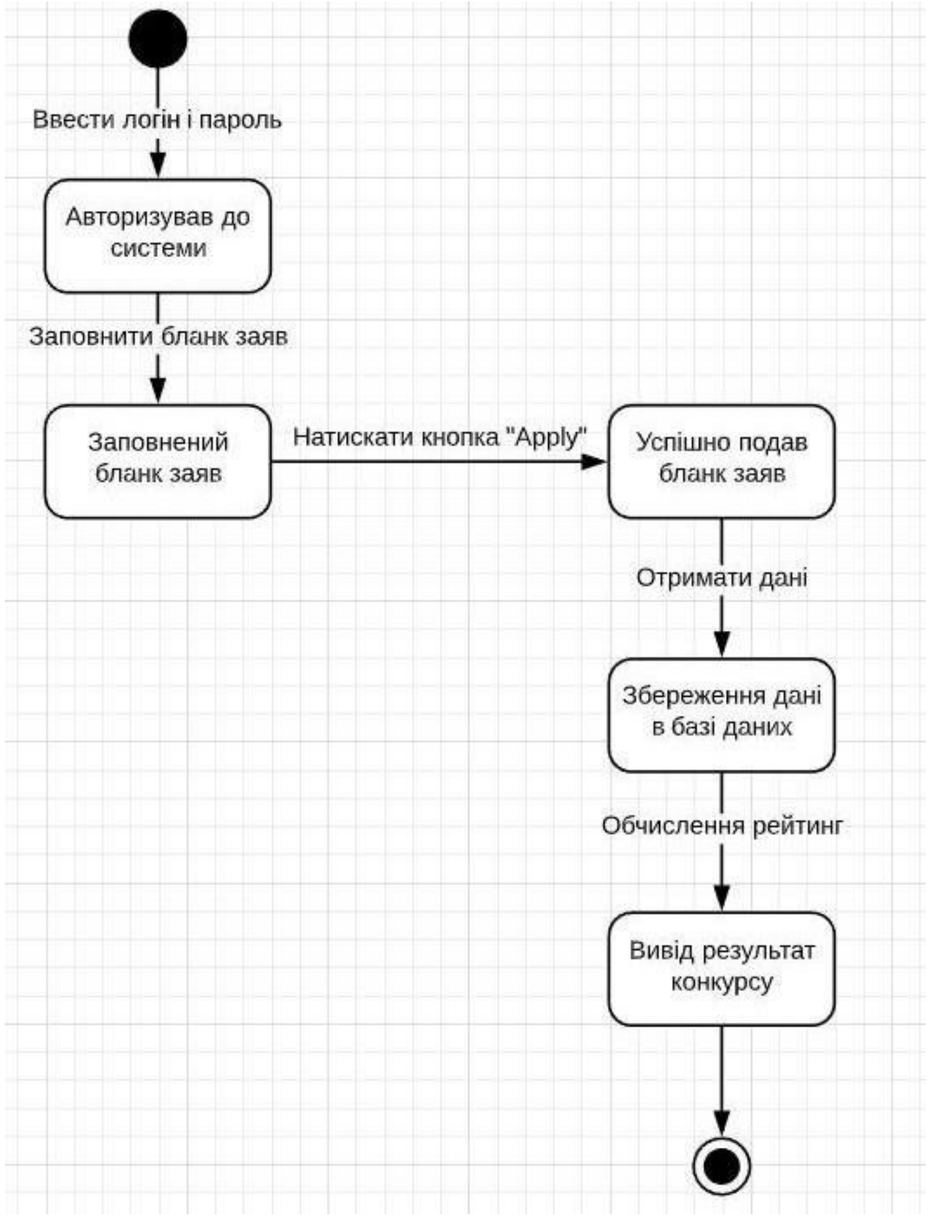


					ДП ІС-6324.02.000.ССВ	
					Схема структурна варіантів використань	Лит. / Маса / Масштаб
Зм.	Арк.	докум.	Підп.	Дата		Аркуш 1 / Аркушів 1
Розроб.		Фам С. Х.			Система підтримки навчання студентів – учасників європейських програм мобільності	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-63
Перев.		Тєлишева Т. О.				
Т. Кон.						
Н. Кон.		Проскура С. Л.				
Затв.		Тєлишева Т. О.				

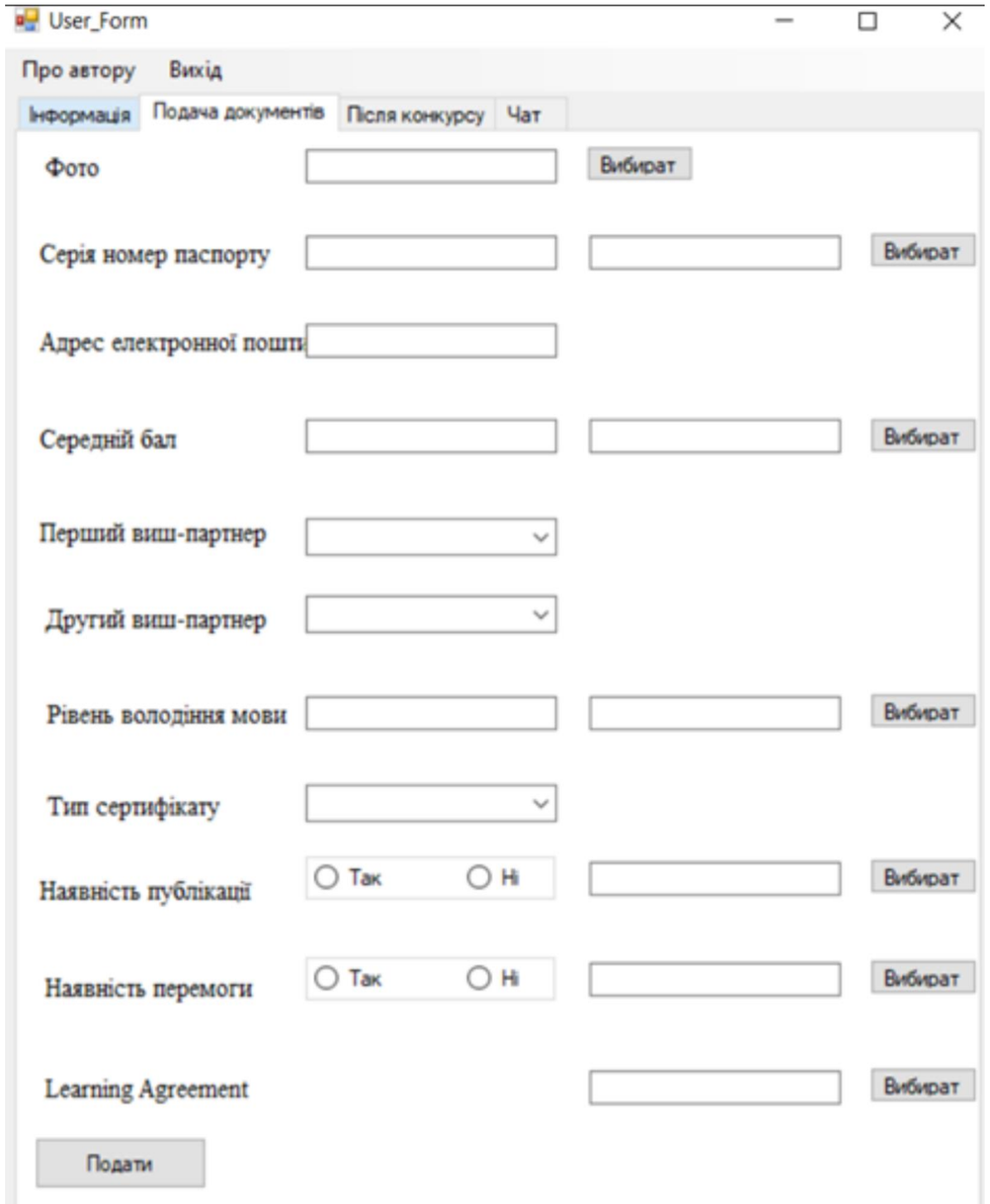
ДП ІС-6324.05.000.ССК



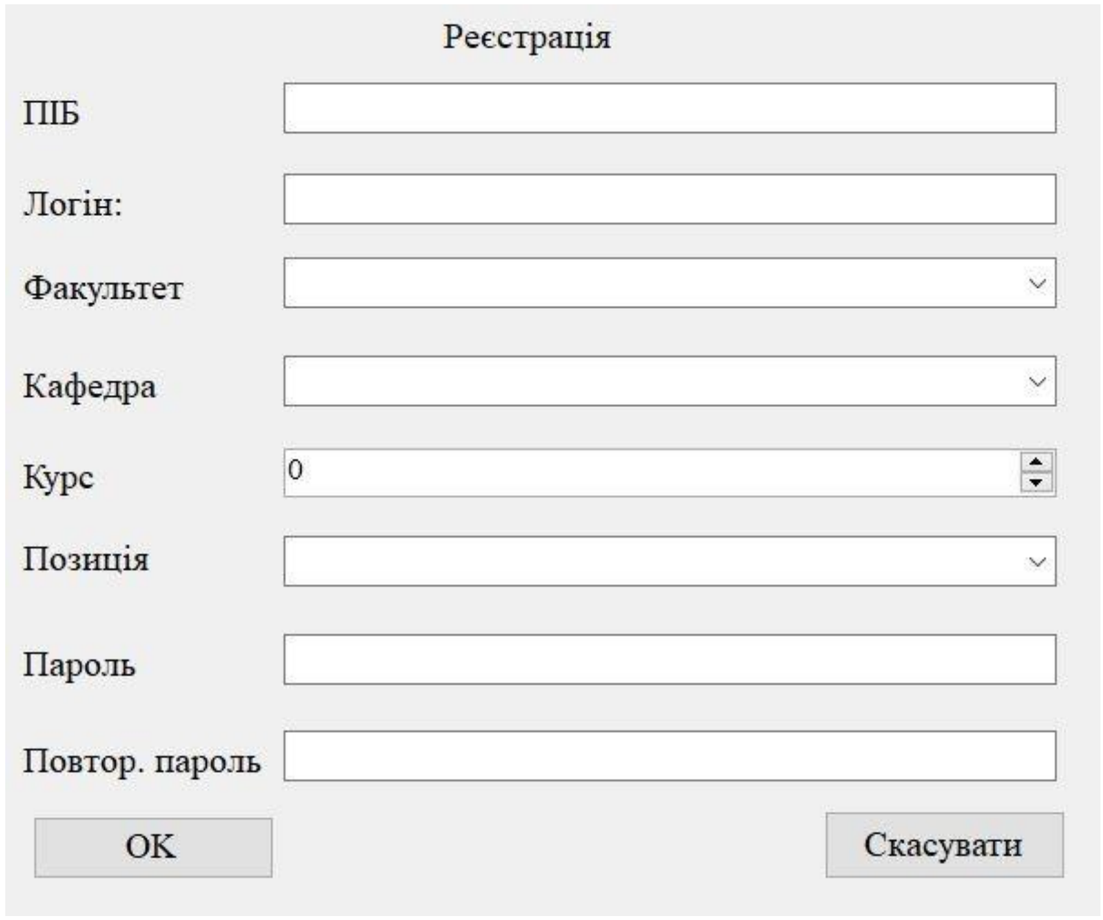
Зм.	Арк.	докум.	Підп.	Дата	ДП ІС-6324.05.000.ССК			
Розроб.		Фам С. Х.			Схема структурна класів програмного забезпечення	Лит.	Маса	Масштаб
Перев.		Тєлишева Т. О.						
Т. Кон.						Аркуш 1	Аркушів 1	
Н. Кон.		Проскура С. Л.			Система підтримки навчання студентів – учасників європейських програм мобільності	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-63		
Затв.		Тєлишева Т. О.						



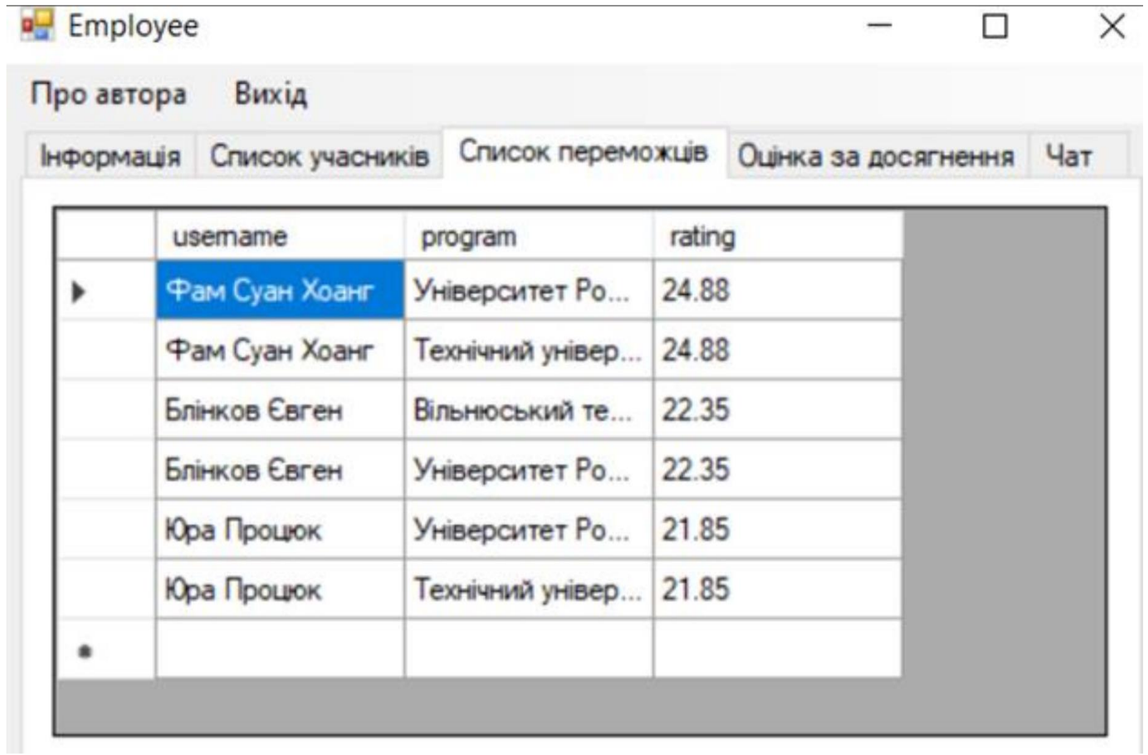
					ДП ІС-6324.03.000.ССС		
Зм.	Арк.	докум.	Підп.	Дата	Схема структурна станів системи	Лит.	Маса / Масштаб
Розроб.		Фам С. Х.					
Перев.		Тєлишева Т. О.				Аркуш 1	Аркушів 1
Т. Кон.							
Н. Кон.		Проскура С. Л.			Система підтримки навчання студентів – учасників європейських програм мобільності	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-63	
Затв.		Тєлишева Т. О.					



					ДП ІС-6324.06.000.КЕ		
Зм.	Арк.	докум.	Підп.	Дата	Креслення вигляду екранних форм	Лит.	Маса / Масштаб
Розроб.		Фам С. Х.					
Перев.		Тєлишева Т. О.				Аркуш 1	Аркушів 1
Т. Кон.							
Н. Кон.		Проскура С. Л.			Система підтримки навчання студентів – учасників європейських програм мобільності	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-63	
Затв.		Тєлишева Т. О.					



					ДП ІС-6324.06.000.КЕ	
Зм.	Арк.	докум.	Підп.	Дата		
Розроб.		Фам С. Х.			Креслення вигляду екранних форм	Лит. / Маса / Масштаб
Перев.		Тєлишева Т. О.				Аркуш 1 / Аркушів 1
Т. Кон.					Система підтримки навчання студентів – учасників європейських програм мобільності	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-63
Н. Кон.		Проскура С. Л.				
Затв.		Тєлишева Т. О.				



					ДП ІС-6324.06.000.КЕ		
Зм.	Арк.	докум.	Підп.	Дата	Креслення вигляду екранних форм	Лит.	Маса / Масштаб
Розроб.		Фам С. Х.					
Перев.		Телишева Т. О.				Аркуш 1	Аркушів 1
Т. Кон.					Система підтримки навчання студентів – учасників європейських програм мобільності	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-63	
Н. Кон.		Проскура С. Л.					
Затв.		Телишева Т. О.					